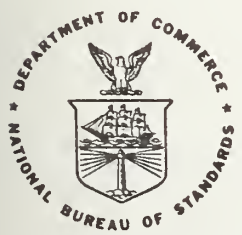
NBSIR 78-1420-3

NBS Minimal BASIC Test Programs - Version 1 User's Manual

Volume 3 - Control Statements, Data Structure, Program Input

David E. Gilsinn
Charles L. Sheppard

Systems and Software Division
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234



U.S. DEPARTMENT OF COMMERCE

NATIONAL BUREAU OF STANDARDS

NBSIR 78-1420-3

NBS MINIMAL BASIC TEST PROGRAMS - VERSION 1 USER'S MANUAL

Volume 3 - Control Statements, Data Structure, Program Input

David E. Gilsinn
Charles L. Sheppard

Systems and Software Division
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234

U.S. DEPARTMENT OF COMMERCE, Juanita M. Kreps, *Secretary*

Dr. Sidney Harman, *Under Secretary*

Jordan J. Baruch, *Assistant Secretary for Science and Technology*

NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Acting Director*

ABSTRACT

This volume is the third of four volumes that comprise the user's guide to the NBS Minimal BASIC test programs. The programs test whether a BASIC processor accepts the syntactical forms and produces semantically meaningful results according to the specifications given in BSR X3.60 Proposed American National Standard for Minimal BASIC. The object of this volume is to complete the testing of the control structures, introduce new data structures, and test the user interactive capability of the language. There are sixty individual programs in this volume that cover looping structures, array variables, exception tests, subroutines, multiway branch structures, data declarations and interactive data inputs. The entire set of programs is available on tape.

Key Words: BASIC, BASIC standard, BASIC validation, compiler validation, computer programming language, computer standards

Table of Contents

	Page
0.0 Introduction	1
34.0 The FOR - NEXT Statements	3
Program Listing	5
Sample Output	9
35.0 Exiting from FOR - Blocks	12
Program Listing	12
Sample Output	13
36.0 Syntax Diagnostic - A FOR - Statement Without a Matching NEXT - Statement	15
Program Listing	15
Sample Output	16
37.0 Syntax Diagnostic - A NEXT - Statement Without a Matching FOR - Statement	17
Program Listing	17
Sample Output	17
38.0 Semantic Error - The Interleaving of Two FOR - Blocks	19
Program Listing	19
Sample Output	20
39.0 Introducing the Dimension Statement	21
Program Listing	23
Sample Output	25
40.0 Extending IF - THEN Capabilities by Using One - Dimensional Arrays in the Comparison	27
Program Listing	27
Sample Output	31
41.0 Extending IF - THEN Capabilities by Using Two - Dimensional Arrays in the Comparison	33
Program Listing	33
Sample Output	38
42.0 The ABS Function With Subscripted Variables for Arguments	39
Program Listing	39
Sample Output	40
43.0 Using Elementary Operations on Subscripted Variables Assigned Same Type Constants	42
Program Listing	42
Sample Output	45
44.0 Using Elementary Operations on Subscripted Variables Assigned Same Type Constants (Continued)	48
Program Listing	48
Sample Output	52
45.0 Using Elementary Operations on Subscripted Variables Assigned Mixed Type Constants	56
Program Listing	56

Sample Output..... 58

46.0 Using Elementary Operations on Subscripted Variables Assigned Mixed Type Constants (Continued)..... 60
Program Listing..... 60
Sample Output..... 62

47.0 Using Elementary Operations on Subscripted Variables Assigned Mixed Type Constants (Continued)..... 64
Program Listing..... 64
Sample Output..... 70

48.0 Addition of More Than Two Terms Containing Array Elements..... 73
Program Listing..... 73
Sample Output..... 75

49.0 Multiplication of More Than Two Terms..... 77
Program Listing..... 77
Sample Output..... 79

50.0 Hierarchy of Operators and Parentheses..... 81
Program Listing..... 81
Sample Output..... 86

51.0 Evaluation of Expressions that have a Variety of Operators..... 89
Program Listing..... 89
Sample Output..... 92

52.0 Exception Test - Zero Raised to a Negative Power..... 94
Program Listing..... 94
Sample Output..... 95

53.0 Exception Test - A Negative Number Raised to a Non - Negative Power..... 96
Program Listing..... 96
Sample Output..... 97

54.0 Semantic Error - Subscripted Variable with Different Numbers of Subscripts..... 98
Program Listing..... 98
Sample Output..... 99

55.0 Exception Test - A Subscript is not in the Range of the Implicit Dimensioning Bounds..... 100
Program Listing..... 100
Sample Output..... 101

56.0 Exception Test - A subscript is not in the Range of an Explicitly Dimensioned Variable..... 102
Program Listing..... 102
Sample Output..... 102

57.0 Attempting String Overflow by Variable Assignment..... 104
Program Listing..... 104
Sample Output..... 105

58.0 Test for Undefined Variables..... 107

Program Listing..... 107
Sample Output..... 108

59.0 Exception Test - On Division by Zero..... 110
Program Listing..... 110
Sample Output..... 111

60.0 Exception Test - On Expression Evaluation Resulting in Overflow..... 113
Program Listing..... 113
Sample Output..... 114

61.0 Semantic Test - On the Magnitude of a Nonzero Numeric Constant That is too Small..... 117
Program Listing..... 117
Sample Output..... 118

62.0 Exception Test - On the Magnitude of a Nonzero Numeric Constant That is too large..... 119
Program Listing..... 119
Sample Output..... 120

63.0 DIM Statement With the OPTION Statement..... 122
Program Listing..... 122
Sample Output..... 123

64.0 Using the OPTION BASE - Statement to Change Implicit Array Lower Bounds..... 125
Program Listing..... 125
Sample Output..... 125

65.0 Testing the Assignment of Zero for an Expression Causing Underflow upon Evaluation..... 127
Program Listing..... 127
Sample Output..... 127

66.0 GOSUB/RETURN - Statement..... 129
Program Listing..... 130
Sample Output..... 131

67.0 Semantic Error - Test on GOSUB Transfer to an Illegal Line Number..... 133
Program Listing..... 133
Sample Output..... 134

68.0 Exception Test - RETURN - Statement Without GOSUB..... 135
Program Listing..... 135
Sample Output..... 136

69.0 Testing Roundoff to Six Significant Digits of Constants of Arbitrary Length..... 137
Program Listing..... 137
Sample Output..... 141

70.0 The ON - GOTO Statement..... 143
Program Listing..... 144
Sample Output..... 147

71.0 Semantic Diagnostic - ON - GOTO Statement Referring to a Non-Existent Line Number..... 148
Program Listing..... 148
Sample Output..... 149

72.0 Exception Test - Value of ON - GOTO Expression Less than One.. 150
Program Listing..... 150
Sample Output..... 150

73.0 Exception Test - Value of ON - GOTO Expression Greater than the Number of Line Numbers in the List..... 152
Program Listing..... 152
Sample Output..... 152

74.0 READ/DATA Statements..... 154
Program Listing..... 156
Sample Output..... 159

75.0 Exception Test - READ - Statement Encounters Insufficient DATA..... 161
Program Listing..... 161
Sample Output..... 162

76.0 Exception Test - Non-Matching String Datum Assigned to a Numeric Variable..... 163
Program Listing..... 163
Sample Output..... 163

77.0 Exception Test - Attempting a String Datum Overflow..... 165
Program Listing..... 165
Sample Output..... 166

78.0 Semantic Interpretation - A Numeric Value in a DATA List Causes an Underflow..... 167
Program Listing..... 167
Sample Output..... 167

79.0 Exception Test - A Numeric Value in a DATA Statement Causes an Overflow..... 169
Program Listing..... 169
Sample Output..... 169

80.0 Exception Test - Overflow Caused by a Numeric Value in a DATA Statement (Continued)..... 171
Program Listing..... 171
Sample Output..... 171

81.0 Restoring READ Data..... 173
Program Listing..... 173
Sample Output..... 174

82.0 INPUT Statement for Numeric Constants..... 176
Program Listing..... 177
Sample Output..... 181

83.0 INPUT of Numeric Data to Subscripted Variables and Unquoted Strings..... 183

Program Listing..... 183
Sample Output..... 187

84.0 Inputting Mixed Data..... 190
Program Listing..... 190
Sample Output..... 194

85.0 Exception Test - Type of Datum Incorrect..... 196
Program Listing..... 196
Sample Output..... 197

86.0 Exception Test - Too much Data in DATA List..... 198
Program Listing..... 198
Sample Output..... 199

87.0 Exception Test - Insufficient Data in DATA List..... 200
Program Listing..... 200
Sample Output..... 201

88.0 Numeric Underflow on INPUT..... 202
Program Listing..... 202
Sample Output..... 202

89.0 Exception Test - Numeric Overflow..... 204
Program Listing..... 204
Sample Output..... 205

90.0 Testing the INT and SGN Functions..... 206
Program Listing..... 206
Sample Output..... 208

91.0 Printing Strings Beyond the Margin..... 210
Program Listing..... 211
Sample Output..... 214

92.0 Tabbing Strings Beyond the Margin..... 217
Program Listing..... 217
Sample Output..... 220

93.0 Exception Test - String Overflow..... 229
Program Listing..... 229
Sample Output..... 230

0.0 INTRODUCTION

This volume is the third in a set of four volumes that comprise the user's guide to the NBS Minimal BASIC test programs. There are sixty individual programs in this volume that cover looping structures, array variables, exception tests, subroutines, multiway branch structures, data declarations and the interactive data inputs. As in the previous volumes the user is assumed to be familiar with the American National Standard for Minimal BASIC, BSR X3.60.

The first tests execute various forms of FOR - NEXT statements. These include tests that use loops with and without the step clause. Although the standard does not specify the depth to which loops can be nested, one test executes a nest of three deep to accomodate at least the looping needed to handle a matrix within some iterative algorithms consisting of one loop. The matrix itself is doubly dimensioned. Another test checks the control variable on exiting from a loop. There are finally some error detection routines for looping.

A natural extension to the looping tests is the introduction of dimensioned variables. These programs first test implicit and explicit dimensioning without the OPTION statement. The OPTION statement in Minimal BASIC allows the user to redefine the lower bound of array indices, which is assumed to be 0 unless altered by the OPTION BASE statement. Next, arrays are used in simple expressions which control some conditional branches. These branch statements are then used to test more complex arithmetic expressions that make use of arrays. Many of the tests of arithmetic expression evaluation are similar to previous arithmetic tests, but in the present volume the expressions also use arrayed variables as well as simple variables and constants. There are a number of exception tests included, as for example, checking for subscripts out of bounds. The OPTION statement is then introduced and tested by checking whether out of bounds errors are detected when the zero-th element of an array is called for after the OPTION BASE was used to specify a lower array bound of one, for example.

Up to this point only the elementary direct transfer GOTO and conditional branch IF-THEN have been tested and used. Two new control structures are now introduced. The tests first examine the GOSUB and RETURN statements. Although no minimal depth of subroutine nesting is specified, one of the tests assumes the capability of handling at least four GOSUB levels. There are also some exception tests associated with the GOSUB capability. The next control structure introduced after the GOSUB statement is the ON-GOTO statement. The main issue in this statement is whether the expression used after the ON is rounded by the test system to the nearest integer rather than truncated. Several tests execute different cases for this statement type, including diagnostic and exception tests.

The final statement types tested in this volume allow insertion of data into a program either as a list or through an external media. The first of these is the DATA statement. Several issues are tested in these routines. For example not only does the READ statement have to assign the proper sequential datum to the READ list item but it must be able to detect whether the datum is compatible with the variable. String data can be specified in two ways. Each of these has to be examined. Such exceptions as whether there is insufficient or too much data have to be tested. Finally the

RESTORE statement has to be tested in order to determine that the data list can be reread. Although the DATA statement is one way of entering data to a program, another way is by means of the INPUT statement that calls for interaction with an external data source. Again the tests not only had to test whether data could be entered but whether it was compatible with what the program assumes is being entered, whether there is too much or too little data being entered, or whether leading and trailing string spaces are accepted or ignored.

The last tests in this volume consider what happens when strings are too long to fit within the margin. The tests include ones to determine how the system handles the tabbing of a string beyond the margin. In general the system must determine how many margin widths fit within the number of spaces requested, skip that many lines and then print the item in the appropriate column computed by a specified formula. The last test uses the INPUT capability to test the exception handling capability of the test system when string overflow is encountered.

34.0 THE FOR-NEXT STATEMENTS

This unit tests several uses of the FOR-statement and the NEXT-statement. They provide for the construction of loops, if the following conditions are met: (1) the control variable is any simple numeric variable; and, (2) both FOR and NEXT have the same control variable. In the absence of the STEP clause, the increment is always +1.

The sequence of statements from the FOR-statement and NEXT-statement forms a block referred to as the FOR-block. FOR-blocks can be nested (one can be contained within another), but they cannot be interleaved. All FOR-blocks are inactive at the initiation of a program but become active upon execution of the FOR-statement. It remains active until it is exited via its NEXT-statement, or until control is transferred to a FOR-statement (which may or may not be the one associated with that FOR-block) having the same control variable. However, control can exit a FOR-block via a control statement in which case the FOR-block should remain active. When exit from a FOR-block is via a NEXT-statement, the value of the control variable should be the first value not used. For the precise specifications a user is referred to section 11 of BSR X3.60.

34.1 FOR/NEXT, Without a STEP Clause

The objective of this subsection is to verify that in the absence of a STEP clause in a FOR-statement, the implementation will assume the increment to be +1.

34.1.1 Initial-Value and Limit Are Integers

The objective here is to initiate the use of the FOR-statement by using integer values only.

34.1.1.1 Different Initial and Limit Values

In this test each loop of the FOR-block is counted, and the final value of the counter, C, determines whether the test passed or failed. There is also a variable, T, in the FOR-block that keeps a running total of the values assigned to the control variable,i.e. I in this case. On output there should be a message indicating whether the test failed or passed. If the test failed then the following message should be printed: TEST FAILED. If the test passed then the following message should be printed: TEST PASSED.

34.1.1.2 Equal Initial and Limit Values

This test shows that looping should not terminate until an increment causes the value of the control variable to exceed the value of the limit, unless there is an exit via a control statement. The actions of the C and the T variables are the same as in section 34.1.1.1 and so is the output.

34.1.2 Fractions Contained in the Limit

The purpose of this test is to continue loop testing in the absence of the STEP clause increments, but using numbers in the control variable limits with fractional values. To have a limit containing a fraction with an integer initial value means that incrementing should cause the control variable to be either less than or greater than the limit, thus never

reaching the limit in the absence of the STEP clause. The actions of the C and the T variables are the same as in test 34.1.1.1, and so is the output.

34.2 FOR/NEXT Using Step Clause

The objective here is to use the STEP clause as a parameter in the FOR-statement.

34.2.1 Using Fractional Increments

This test verifies that the processor recognizes and, within machine accuracy considerations, will increment a loop control variable with a fractional step in the proper manner.

34.2.1.1 For an Increasing Control Value

The STEP clause for this test is a fraction while its other parameters are integers. The C and T variables are again used. The T variable for this test is testing for added increments of +.5, since +.5 is the value for the STEP clause in this test. The output for this test should be similar to the output in test 34.1.1.1.

34.2.1.2 For a Decreasing Control Value

The object of this test is to show that looping of a FOR-block, when the initial value is to be decreased in value, should not terminate until the control variable has been assigned a value less than the value of the limit, unless there is an exit via a control statement. As in the previous test, C and T variables are used. The T variable for this test is testing for added decrements of -.5, since -.5 is the value for the STEP clause in this test. The output for this test should be the same as the output in test 34.1.1.1.

34.2.2 Using Integer Increments

The objective of this test is to use the STEP clause for increments where the assigned STEP parameters are integer valued.

34.2.2.1 For Decreasing the Control Value

This test shows some of the various ways in which the STEP clause can be used to decrease the value of the initial value to a desired limit value. The output for all tests below is similar to test 34.1.1.1.

34.2.2.1.1 Positive to Positive

The object here is to test decreasing the control variable's value from a positive number to a smaller positive number. The value for the STEP clause in this test is -2.

34.2.2.1.2 Positive to Negative

The object here is to test decreasing the initial value from a positive number to a negative number in step increments of -4.

34.2.2.1.3 Negative to Negative

The object here is to test decreasing the initial-value from one negative number to another negative number in increments of -1.

34.2.2.2 For Increasing Control Variable

This test is constructed to increase the control variable's value from one negative number to another negative number in increments of 2.

34.2.3 Fractions

The object of this test is to show that the initial value is allowed to contain a fraction. The control variable will be stepped forward each time by +2.

34.3 Nesting FOR-Blocks, Three Deep

The object of this test is to test that FOR-blocks can be nested to a depth of three. That is, three FOR-blocks can be active at one time.

```
**********
* PROGRAM FILE 34 *
**********
```

```
0010 PRINT "PROGRAM FILE 34"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                SECTION 34.1: FOR/NEXT, WITHOUT STEP CLAUSE.
0100 PRINT
0110 PRINT "     SECTION 34.1.1: INITIAL VALUE AND LIMIT ARE INTEGERS.
0120 PRINT
0130 PRINT "          SECTION 34.1.1.1: DIFFERENT VALUES, LOW TO HIGH.
0140 PRINT
0150 PRINT "                                  BEGIN TEST."
0160 PRINT
0170 LET C=0
0180 LET T=0
0190 FOR I=-2 TO 3
0200 LET C=C+1
0210 LET T=T+I
0220 NEXT I
0230 IF C<>6 THEN 250
0240 IF T=3 THEN 270
0250 PRINT "                    THE 6 LOOPS (-2 TO 3), FAILED TEST."
0260 GOTO 300
0270 PRINT "                    THE 6 LOOPS (-2 TO 3), PASSED TEST."
0280 PRINT
0290 PRINT "                                    END TEST."
0300 PRINT
```

```
0310 PRINT "                          SECTION 34.1.1.2: EQUAL VALUES."
0320 PRINT
0330 PRINT "                                   BEGIN TEST."
0340 PRINT
0350 LET C=0
0360 LET T=0
0370 FOR I=3 TO 3
0380 LET C=C+1
0390 LET T=T+I
0400 NEXT I
0410 IF C<>1 THEN 430
0420 IF T=3 THEN 450
0430 PRINT "                     THE 1 LOOP (3 TO 3), FAILED TEST."
0440 GOTO 480
0450 PRINT "                     THE 1 LOOP (3 TO 3), PASSED TEST."
0460 PRINT
0470 PRINT "                                   END TEST."
0480 PRINT
0490 PRINT "           SECTION 34.1.2: FRACTION CONTAINED IN THE LIMIT.
0500 PRINT
0510 PRINT
0515 LET C=0
0520 LET T=0
0530 FOR I=4 TO 5.9
0540 LET C=C+1
0550 LET T=T+I
0560 NEXT I
0570 IF C<>2 THEN 590
0580 IF T=9 THEN 610
0590 PRINT "                     THE 2 LOOPS (4 TO 5.9), FAILED TEST."
0600 GOTO 640
0610 PRINT "                     THE 2 LOOPS (4 TO 5.9), PASSED TEST."
0620 PRINT
0630 PRINT "                                   END TEST."
0640 PRINT
0650 PRINT "                SECTION 34.2: FOR/NEXT, USING STEP CLAUSE."
0660 PRINT
0670 PRINT "               SECTION 34.2.1: USING FRACTIONAL INCREMENTS.
0680 PRINT
0690 PRINT "            SECTION 34.2.1.1: FOR INCREASING INITIAL VALUE.
0700 PRINT
0710 PRINT "                                   BEGIN TEST."
0720 PRINT
0730 LET C=0
0740 LET T=0
0750 FOR I=0 TO 4 STEP .5
0760 LET C=C+1
0770 LET T=T+I
0780 NEXT I
0790 IF C<>9 THEN 810
0800 IF T=18 THEN 830
0810 PRINT "   THE 9 LOOPS (0 TO 4), INCREMENTED BY .5, FAILED TEST."
0820 GOTO 860
0830 PRINT "   THE 9 LOOPS (0 TO 4), INCREMENTED BY .5, PASSED TEST."
0840 PRINT
0850 PRINT "                                   END TEST."
0860 PRINT
```

```
0870 PRINT "           SECTION 34.2.1.2: FOR DECREASING INITIAL VALUE.
0880 PRINT
0890 PRINT "                                   BEGIN TEST."
0900 PRINT
0910 LET C=0
0920 LET T=0
0930 FOR I=2 TO 2 STEP -.5
0940 LET C=C+1
0950 LET T=T+I
0960 NEXT I
0970 IF C<>1 THEN 990
0980 IF T=2 THEN 1010
0990 PRINT "   THE 1 LOOP (2 TO 2), INCREMENTED BY -.5, FAILED TEST."
1000 GOTO 1040
1010 PRINT "   THE 1 LOOP (2 TO 2), INCREMENTED BY -.5, PASSED TEST."
1020 PRINT
1030 PRINT "                                     END TEST."
1040 PRINT
1050 PRINT "                SECTION 34.2.2: USING INTEGER INCREMENTS."
1060 PRINT
1070 PRINT "             SECTION 34.2.2.1: FOR DECREASING INITIAL VALUE.
1080 PRINT
1090 PRINT "                 SECTION 34.2.2.1.1: POSITIVE TO POSITIVE."
1100 PRINT
1110 PRINT "                                   BEGIN TEST."
1120 LET C=0
1130 LET T=0
1140 FOR I=4 TO 1 STEP -2
1150 LET C=C+1
1160 LET T=T+I
1170 NEXT I
1180 IF C<>2 THEN 1200
1190 IF T=6 THEN 1220
1200 PRINT "   THE 2 LOOPS (4 TO 1), INCREMENTED BY -2, FAILED TEST."
1210 GOTO 1250
1220 PRINT "   THE 2 LOOPS (4 TO 1), INCREMENTED BY -2, PASSED TEST."
1230 PRINT
1240 PRINT "                                     END TEST."
1250 PRINT
1260 PRINT "                 SECTION 34.2.2.1.2: POSITIVE TO NEGATIVE."
1270 PRINT
1280 PRINT "                                   BEGIN TEST."
1290 PRINT
1300 LET C=0
1310 LET T=0
1320 FOR I=8 TO -8 STEP -4
1330 LET C=C+1
1340 LET T=T+I
1350 NEXT I
1360 IF C<>5 THEN 1380
1370 IF T=0 THEN 1400
1380 PRINT "   THE 5 LOOPS (8 TO -8), INCREMENTED BY -4, FAILED TEST."
1390 GOTO 1430
1400 PRINT "   THE 5 LOOPS (8 TO -8), INCREMENTED BY -4, PASSED TEST."
1410 PRINT
1420 PRINT "                                     END TEST."
1430 PRINT
```

```
1440 PRINT "                SECTION 34.2.2.1.3: NEGATIVE TO NEGATIVE."
1450 PRINT
1460 PRINT "                                    BEGIN TEST."
1470 PRINT
1480 LET C=0
1490 LET T=0
1500 FOR I=-1 TO -3 STEP -1
1510 LET C=C+1
1520 LET T=T+I
1530 NEXT I
1540 IF C<>3 THEN 1560
1550 IF T=(-6) THEN 1580
1560 PRINT "   THE 3 LOOPS (-1 TO -3), INCREMENTED BY -1, FAILED TEST.
1570 GOTO 1610
1580 PRINT "   THE 3 LOOPS (-1 TO -3), INCREMENTED BY -1, PASSED TEST.
1590 PRINT
1600 PRINT "                                     END TEST."
1610 PRINT
1620 PRINT "        SECTION 34.2.2.2: FOR INCREASING INITIAL VALUE."
1630 PRINT
1640 PRINT "                                    BEGIN TEST."
1650 PRINT
1660 LET C=0
1670 LET T=0
1680 FOR I=-12 TO -5 STEP 3
1690 LET C=C+1
1700 LET T=T+I
1710 NEXT I
1720 IF C<>3 THEN 1740
1730 IF T=(-27) THEN 1760
1740 PRINT "   THE 3 LOOPS (-12 TO -5), INCREMENTED BY 3, FAILED TEST.
1750 GOTO 1790
1760 PRINT "   THE 3 LOOPS (-12 TO -5), INCREMENTED BY 3, PASSED TEST.
1770 PRINT
1780 PRINT "                                     END TEST."
1790 PRINT
1800 PRINT " SECTION 34.2.3: FRACTION CONTAINED BY THE INITIAL VALUE."
1810 PRINT
1820 PRINT "                                    BEGIN TEST."
1830 PRINT
1840 LET C=0
1850 LET T=0
1860 FOR I=1.5 TO 3 STEP 2
1870 LET C=C+1
1880 LET T=T+I
1890 NEXT I
1900 IF C<>1 THEN 1920
1910 IF T=1.5 THEN 1940
1920 PRINT "   THE 1 LOOP (1.5 TO 3), INCREMENTED BY 2, FAILED TEST."
1930 GOTO 1970
1940 PRINT "   THE 1 LOOP (1.5 TO 3), INCREMENTED BY 2, PASSED TEST."
1950 PRINT
1960 PRINT "                                     END TEST."
1970 PRINT
1980 PRINT "            SECTION 34.3: NESTING FOR-BLOCKS, THREE DEEP."
1990 PRINT
2000 PRINT "                                    BEGIN TEST."
```

```
2010 PRINT
2020 LET C=0
2030 LET T=0
2040 FOR I1=1 TO 2
2050 FOR I2=3 TO 1 STEP -1
2060 FOR I3=1 TO 3 STEP 1
2120 LET C=C+1
2130 LET T=T+I3
2190 NEXT I3
2200 NEXT I2
2210 NEXT I1
2220 IF C<>18 THEN 2240
2230 IF T=36 THEN 2260
2240 PRINT "                          3 NESTED LOOPS, FAILED TEST."
2250 GOTO 2290
2260 PRINT "                          3 NESTED LOOPS, PASSED TEST."
2270 PRINT
2280 PRINT "                                   END TEST."
2290 PRINT
2300 PRINT
2310 END
```

```
*****************
* SAMPLE OUTPUT *
*****************
```

```
PROGRAM FILE 34

            SECTION 34.1: FOR/NEXT, WITHOUT STEP CLAUSE.

     SECTION 34.1.1: INITIAL VALUE AND LIMIT ARE INTEGERS.

          SECTION 34.1.1.1: DIFFERENT VALUES, LOW TO HIGH.

                         BEGIN TEST.

                THE 6 LOOPS (-2 TO 3), PASSED TEST.

                         END TEST.

               SECTION 34.1.1.2: EQUAL VALUES.

                         BEGIN TEST.

                THE 1 LOOP (3 TO 3), PASSED TEST.

                         END TEST.
```

SECTION 34.1.2: FRACTION CONTAINED IN THE LIMIT.

THE 2 LOOPS (4 TO 5.9), PASSED TEST.

END TEST.

SECTION 34.2: FOR/NEXT, USING STEP CLAUSE.

SECTION 34.2.1: USING FRACTIONAL INCREMENTS.

SECTION 34.2.1.1: FOR INCREASING INITIAL VALUE.

BEGIN TEST.

THE 9 LOOPS (Ø TO 4), INCREMENTED BY .5, PASSED TEST.

END TEST.

SECTION 34.2.1.2: FOR DECREASING INITIAL VALUE.

BEGIN TEST.

THE 1 LOOP (2 TO 2), INCREMENTED BY -.5, PASSED TEST.

END TEST.

SECTION 34.2.2: USING INTEGER INCREMENTS.

SECTION 34.2.2.1: FOR DECREASING INITIAL VALUE.

SECTION 34.2.2.1.1: POSITIVE TO POSITIVE.

BEGIN TEST.

THE 2 LOOPS (4 TO 1), INCREMENTED BY -2, PASSED TEST.

END TEST.

SECTION 34.2.2.1.2: POSITIVE TO NEGATIVE.

BEGIN TEST.

THE 5 LOOPS (8 TO -8), INCREMENTED BY -4, PASSED TEST.

END TEST.

SECTION 34.2.2.1.3: NEGATIVE TO NEGATIVE.

BEGIN TEST.

THE 3 LOOPS (-1 TO -3), INCREMENTED BY -1, PASSED TEST.

END TEST.

SECTION 34.2.2.2: FOR INCREASING INITIAL VALUE.

BEGIN TEST.

THE 3 LOOPS (-12 TO -5), INCREMENTED BY 3, PASSED TEST.

END TEST.

SECTION 34.2.3: FRACTION CONTAINED BY THE INITIAL VALUE.

BEGIN TEST.

THE 1 LOOP (1.5 TO 3), INCREMENTED BY 2, PASSED TEST.

END TEST.

SECTION 34.3: NESTING FOR-BLOCKS, THREE DEEP.

BEGIN TEST.

3 NESTED LOOPS, PASSED TEST.

END TEST.

35.0 EXITING FROM FOR-BLOCKS

35.1 FOR-Block Exiting Via Control Statement

This routine tests exiting from a FOR-block via a control statement. In this routine the control statement is the IF-THEN-statement. Since the first FOR-block for this test does not exit naturally (via its NEXT-statement), there should be only 11 loops performed and a T variable sum of 30. The output is similar to test 34.1.1.1.

35.2 Compatability Between Initial-Value, Limit, and STEP Clause

This routine tests the compatibility between the initial value, the limit and the increment. There are two cases which determine this compatibility: (1) if the initial value is smaller than the limit, then the increment's value must be positive; and (2) if the initial value is larger than the limit, then the increment's value must be negative. If either case is violated, no looping in the FOR-block should be performed. In this test the initial value, the limit, and the increment are not compatible, therefore the number of loops should be zero. The output is similar to test 34.1.1.1.

35.3 Normal Exit Via NEXT-Statement

The object here is to test the value of the control variable upon exiting via the NEXT-statement. Its value should be the first value not used. For output refer to test 34.1.1.1.

```
**********
* PROGRAM FILE 35 *
**********
```

```
0010 PRINT "PROGRAM FILE 35"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "     SECTION 35.1: FOR-BLOCK EXITING VIA CONTROL STATEMENT."
0100 PRINT
0110 PRINT "                                   BEGIN TEST."
0120 PRINT
0130 LET C=0
0140 LET T=0
0150 FOR I=1 TO 10
0160 LET C=C+1
0170 LET T=T+I
0180 IF I=5 THEN 200
0190 NEXT I
0200 IF I=5 THEN 240
0210 PRINT "     THE SYSTEM FAILED TO RETAIN THE INCREMENTED INITIAL"
```

```
0220 PRINT "VALUE VIA A CONTROL STATEMENT."
0230 GOTO 350
0240 FOR J=2 TO 4
0250 FOR I=2 TO 3
0260 LET C=C+1
0270 LET T=T+I
0280 NEXT I
0290 NEXT J
0300 IF C<>11 THEN 320
0310 IF T=30 THEN 340
0320 PRINT "          THE 11 LOOPS (EXIT VIA IF/THEN), FAILED TEST."
0330 GOTO 370
0340 PRINT "          THE 11 LOOPS (EXIT VIA IF/THEN), PASSED TEST."
0350 PRINT
0360 PRINT "                                    END TEST."
0370 PRINT
0380 PRINT " SECTION 35.2 COMPATIBILITY/INITIAL VALUE, LIMIT AND STEP."
0390 PRINT
0400 PRINT "                                  BEGIN TEST."
0410 PRINT
0420 LET C=0
0430 FOR I=4 TO 2 STEP 1
0440 LET C=C+1
0450 NEXT I
0460 IF C=0 THEN 490
0470 PRINT " THE SKIPPING OF (4 TO 2), INCREMENTED BY 1, FAILED TEST."
0480 GOTO 520
0490 PRINT " THE SKIPPING OF (4 TO 2), INCREMENTED BY 1, PASSED TEST."
0500 PRINT
0510 PRINT "                                    END TEST."
0520 PRINT
0530 PRINT "             SECTION 35.3: NORMAL EXIT VIA NEXT-STATEMENT."
0540 PRINT
0550 PRINT "                                  BEGIN TEST."
0560 PRINT
0570 LET C=0
0580 FOR I=1 TO 6
0590 LET C=C+1
0600 NEXT I
0610 IF C<>6 THEN 630
0620 IF I=7 THEN 650
0630 PRINT "CONTROL VALUE OF 7 (EXIT VIA NEXT-STATEMENT), FAILED TEST."
0640 GOTO 680
0650 PRINT "CONTROL VALUE OF 7 (EXIT VIA NEXT-STATEMENT), PASSED TEST."
0660 PRINT
0670 PRINT "                                    END TEST."
0680 PRINT
0690 PRINT
0700 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 35

SECTION 35.1: FOR-BLOCK EXITING VIA CONTROL STATEMENT.

BEGIN TEST.

THE 11 LOOPS (EXIT VIA IF/THEN), PASSED TEST.

END TEST.

SECTION 35.2 COMPATIBILITY/INITIAL VALUE, LIMIT AND STEP.

BEGIN TEST.

THE SKIPPING OF (4 TO 2), INCREMENTED BY 1, PASSED TEST.

END TEST.

SECTION 35.3: NORMAL EXIT VIA NEXT-STATEMENT.

BEGIN TEST.

CONTROL VALUE OF 7 (EXIT VIA NEXT-STATEMENT), PASSED TEST.

END TEST.

36.0 SYNTAX DIAGNOSTIC - A FOR-STATEMENT WITHOUT A MATCHING NEXT-STATEMENT

This routine and the next two perform tests on the FOR-NEXT statement which should be diagnosed as errors. These are specifically constructed to demonstrate the diagnostic capability of the language processor. Although no exceptions have been specified with regard to FOR-NEXT Statements the situations tested here are considered significant and require a processor to report the error.

The objective of this test is to verify that the execution of a FOR-statement without a matching NEXT-statement will be recognized as a syntactic error. This error must be recognized and reported. It should result in the execution of the program being suspended. There should be some form of implementation-defined diagnostic on output.

```
********************
* PROGRAM FILE 36 *
********************
```

```
0010 PRINT "PROGRAM FILE 36"
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "                          SECTION 36.0"
0140 PRINT
0150 PRINT "      A FOR-STATEMENT WITHOUT A MATCHING NEXT-STATEMENT."
0160 PRINT
0170 PRINT
0180 PRINT
0190 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO USE A FOR-STATE-"
0200 PRINT "MENT WITHOUT ANY OCCURRING OR MATCHING NEXT-STATEMENT IN"
0210 PRINT "THE SEQUENTIALLY FOLLOWING PROGRAM LINES.  IF THE SYSTEM"
0220 PRINT "RECOGNIZES THIS AS A SYNTAX ERROR THE TEST PASSED"
0250 PRINT
0260 PRINT
0270 PRINT
0280 PRINT "                                  BEGIN TEST."
0290 PRINT
0300 LET N=0
0310 FOR I=1 TO 100
0320 LET N=N+1
0330 PRINT N;"LOOPS WERE MADE, THEREFORE TEST FAILS SINCE THE EXCLUSION"
0340 PRINT "OF A MATCHING NEXT-STATEMENT WAS NOT RECOGNIZED BY THE SYS-"
0350 PRINT "TEM."
0360 PRINT
0370 PRINT "                                    END TEST."
0380 PRINT
0390 END
```

****
* SAMPLE OUTPUT *
****

In order for this test to pass, an error must be diagnosed and reported. For example, a possible error diagnostic for this program might be:

```
? FOR WITHOUT NEXT IN LINE 310
```

37.0 SYNTAX DIAGNOSTIC - A NEXT-STATEMENT WITHOUT A MATCHING FOR-STATEMENT

The objective of this test is to verify that upon the execution of a program, which contains a NEXT-statement but no matching FOR-statement, the implementation will report a diagnosed error. The test is specifically constructed without a matching FOR-statement for the NEXT-statement in statement 260 of Program File 37. On output, there should be an implementation-specific diagnostic, but it should point to the fact that the NEXT-statement has no associated FOR-statement.

****
* PROGRAM FILE 37 *
****

```
0010 PRINT "PROGRAM FILE 37"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 37.0"
0100 PRINT
0110 PRINT "      A NEXT-STATEMENT WITHOUT A MATCHING FOR-STATEMENT."
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO USE A NEXT-STATE-"
0160 PRINT "MENT WITHOUT ANY OCCURRING OR MATCHING FOR-STATEMENT IN THE"
0170 PRINT "SEQUENTIALLY FOLLOWING PROGRAM LINES.  IF THE SYSTEM RECOG-"
0180 PRINT "NIZES THIS AS A SYNTAX ERROR THE TEST PASSED"
0210 PRINT
0220 PRINT
0230 PRINT
0240 PRINT "                                  BEGIN TEST."
0250 PRINT
0260 NEXT I
0270 PRINT "     TEST FAILS, THE SYSTEM DID NOT RECOGNIZE THE EXCLUSION"
0280 PRINT "OF A MATCHING FOR-STATEMENT."
0290 PRINT
0300 PRINT "                                    END TEST."
0310 PRINT
0320 END
```

****
* SAMPLE OUTPUT *

****

A fatal error diagnostic is required as output. Again, the exact diagnostic is implementation specific, but a possible message might be:

? NEXT WITHOUT FOR IN LINE 260

38.0 SEMANTIC ERROR - THE INTERLEAVING OF TWO FOR-BLOCKS

The objective of this test is to verify that upon the execution of a program which contains two FOR-blocks that are interleaved--i.e., a NEXT-statement is matched with a FOR-statement with a different control variable-- the implementation will diagnose and report an error. The test contains two FOR-blocks that are interleaved by associating FOR-statements at lines 270 and 280 with NEXT-statements at lines 300 and 310 respectively in Program File 38. On output, there should be an implementation-specific diagnostic.

```
**********
* PROGRAM FILE 38 *
**********
```

```
0010 PRINT "PROGRAM FILE 38"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 38.0"
0100 PRINT
0110 PRINT "               THE INTERLEAVING OF TWO FOR-BLOCKS."
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO MATCH A NEXT-STATE-"
0160 PRINT "MENT WITH A FOR-STATEMENT WHICH HAS A DIFFERENT CONTROL"
0170 PRINT "VARIABLE.  IF THE SYSTEM RECOGNIZES THIS AS A SEMANTIC ERROR"
0180 PRINT "THE TEST PASSED"
0210 PRINT
0220 PRINT
0230 PRINT
0240 PRINT "                                  BEGIN TEST."
0250 PRINT
0260 LET N=0
0270 FOR I=1 TO 50
0280 FOR J=1 TO 100
0290 LET N=N+1
0300 NEXT I
0310 NEXT J
0320 PRINT "     SYSTEM FAILED TO RECOGNIZE FOR-NEXT STATEMENTS BEING"
0330 PRINT "INTERLEAVED, THEREFORE, SYSTEM FAILED TEST."
0340 PRINT
0350 PRINT
0360 PRINT "                                   END TEST."
0370 PRINT
0380 END
```

****
* SAMPLE OUTPUT *
****

Again, as in the past two error tests, the system fails if there is no diagnostic given. The following two diagnostic messages are not necessarily ideal, but as a combination do indicate the problem source:

```
?NEXT WITHOUT FOR IN LINE 300
?FOR WITHOUT NEXT IN LINE 270
```

The point of these messages is that the system is not looking for an associated FOR-statement in a higher level block or for a NEXT-statement in a lower level block.

39.0 INTRODUCING THE DIMENSION STATEMENT

The tests in the next several sections are for the dimension-statement which reserves space for arrays, that can be either one or two dimensional. If a dimension-statement is not used to allocate storage, then a default space allocation is made. In section 39.1 this default will be tested. The default means that unless declared otherwise in a dimension-statement, all array subscripts have a lower bound of zero and a upper bound of ten. Therefore, under default there should be space reserved for 11 elements in one-dimensional arrays and 121 elements in two-dimensional arrays. But, by use of a dimension-statement, the subscript(s) of an array may be declared to have an upper bound other than ten, and by use of the OPTION-statement, the subscripts of all arrays may be declared to have a lower bound of either one or zero. The reader is referred to section 15 on array declaration in BSR X3.60 for the specification.

39.1 Implicit Dimensioning

The object here is to verify that implementations recognize default space allocations for arrays.

39.1.1 One-Dimensional Arrays

The object is to test the default space allocation for one-dimensional arrays. In this case, space should be reserved for 11 elements. In the first part of the test, the subscripted variables A(0), A(1), A(2), ..., A(10) of the implicit dimensional array A(I) are assigned the values 0, 1, 2, ..., 10 respectively. In the second part, there is a check made on the assignment of the eleven elements to the array A(I). This is done by the use of a counter, C, and the total sum, A1, of the eleven elements. There should be a count of eleven array elements and the sum of the elements should be 55. On output there should be a message to the following effect, if the test failed: IMPLICIT SINGLE DIMENSIONING TEST, FAILED. If the test passed, then the following message should be printed: IMPLICIT SINGLE DIMENSIONING TEST, PASSED. Another failure of the test could occur if a system diagnostic indicated that an array element could not be accessed. If in fact the system indicates failure to dimension an array, then the system entirely fails implicit single dimensioning.

39.1.2 Two-Dimensional Arrays

The object is to test the default space allocation for two-dimensional arrays. Space should be reserved for 121 elements. In the first part of this test the subscripted variables B(0,0), B(0,1), B(0,2), ..., B(10,10) of the implicit dimensioned array B(I,J) are assigned the sum I+J. In the second part of the test, a check is made on whether space was reserved for the 121 elements of array B(I,J). This is accomplished through the use of a counter, C. The counter keeps track of the number of loops performed by a FOR-block, while the sum of all elements stored in array B(I,J) is computed. The final count for this test should be 121 and the sum of the elements should be 1210.

On output, there should be one of two messages printed. If the test fails, then the following message should be printed: IMPLICIT DOUBLE DIMENSIONING TEST, FAILED. If the test passes then the following message should be printed: IMPLICIT DOUBLE DIMENSIONING TEST, PASSED.

39.2 The Dimension-Statement Without the OPTION-Statement

The object of this test is to verify that the implementation recognizes array declarations by the dimension-statement for both one- and two-dimensional arrays.

39.2.1 Used With One-Dimensional Arrays

The object of this test is to verify that, by use of the dimension-statement, an array can be declared to have an upper bound greater than 10. It would still retain its lower bound of zero. In the first part of the test, the subscripted variables D(0), D(1), D(2), ..., D(20) of array D(I) are assigned the values of 1, 2, 3, ..., 21, respectively. In the second part of this test, the array allocation of 21 elements for D(I) is accessed in order to verify the previous assignment. This is accomplished by use of a loop counter, C and a total sum variable, D1. The number of loop counts for this test should be 21, and the sum of the array elements should be 231.

If the test fails, then the following message should be printed: USE OF DIM FOR SINGLE DIMENSIONING, FAILED TEST. If the test passes, then the following message should be printed: USE OF DIM FOR SINGLE DIMENSIONING, PASSED TEST.

39.2.2 Used With Two-Dimensional Arrays

The object of this test is to verify that, by use of the dimension-statement, two-dimensional arrays can be declared to have upper bounds greater than 10 for each dimension. This is accomplished in three steps. In step one, the first subscript upper bound dimensioned greater than 10 for array N(I,J) while the second subscript remains less than 10. The subscripted variable N(I,1) is assigned the value I+1. The subscripted variables N(0,2), N(1,2), N(2,2), ..., N(20,2) are each assigned the negative value of the first integer of their respective pair of subscripted integers. Finally N(I,0) is assigned the value 1. As a check on the space allocation for each element of array N(20,2), all members of the array are added and the value should be 42. In the second step, an array P(2,20) is dimensioned. The first subscript of the array P(I,J) remains less than 10 while the second subscript is allowed to be greater than 10. The subscripted variables P(1,0), P(1,1), P(1,2), ..., P(2,20) are each assigned a value equal to the product of the subscripts. As a verification of the space allocation, all members of the array P(2,20) are added to each other and their sum should be 630. In the third dimensioning, both the first and second subscripts of the array R(I,J) are simultaneously allowed to have upper bounds greater than 10. For this part of the test, each array element R(I,J) is assigned the sum I+J. The total sum of all the array elements R(I,J) should be 8820.

On output, there should be one of two possible messages. If the test fails then the following message should be printed: DIM FOR DOUBLE DIMENSIONING, FAILED TEST. If the test passes the following message should be printed: DIM FOR DOUBLE DIMENSIONING, PASSED TEST.

****

```
                    * PROGRAM FILE 39 *
                    ********************
```

```
0010 PRINT "PROGRAM FILE 39"
0020 PRINT
0030 PRINT
0040 PRINT
0070 PRINT "                    SECTION 39.1: IMPLICIT DIMENSIONING."
0080 PRINT
0090 PRINT "                  SECTION 39.1.1: ONE-DIMENSIONAL ARRAYS."
0100 PRINT
0110 PRINT "                                  BEGIN TEST."
0120 PRINT
0130 FOR I=0 TO 10
0140 LET A(I)=I
0150 NEXT I
0160 LET A1=0
0170 LET C=0
0180 FOR I=10 TO 0 STEP -1
0185 LET Y=A(I)
0190 LET A1=A1+Y
0200 LET C=C+1
0210 NEXT I
0220 IF C<>11 THEN 240
0230 IF A1=55 THEN 260
0240 PRINT "              IMPLICIT SINGLE DIMENSIONING TEST, FAILED."
0250 GOTO 280
0260 PRINT "              IMPLICIT SINGLE DIMENSIONING TEST, PASSED."
0270 PRINT
0280 PRINT "                                  END TEST."
0290 PRINT
0300 PRINT "                  SECTION 39.1.2: TWO-DIMENSIONAL ARRAYS."
0310 PRINT
0320 PRINT "                                  BEGIN TEST."
0330 PRINT
0340 FOR I=0 TO 10
0350 FOR J=0 TO 10
0355 LET Z=I+J
0360 LET B(I,J)=Z
0370 NEXT J
0380 NEXT I
0390 LET B1=0
0400 LET C=0
0410 FOR I=10 TO 0 STEP -1
0420 FOR J=10 TO 0 STEP -1
0425 LET W=B(I,J)
0430 LET B1=B1+W
0440 LET C=C+1
0450 NEXT J
0460 NEXT I
0470 IF C<>121 THEN 490
0480 IF B1=1210 THEN 510
0490 PRINT "               IMPLICIT DOUBLE DIMENSIONING TEST, FAILED."
```

```
0500 GOTO 530
0510 PRINT "                IMPLICIT DOUBLE DIMENSIONING TEST, PASSED."
0520 PRINT
0530 PRINT "                                    END TEST."
0540 PRINT
0550 PRINT "SECTION 39.2: THE DIM-STATEMENT WITHOUT OPTION-STATEMENT."
0560 PRINT
0570 PRINT "         SECTION 39.2.1: USED WITH ONE-DIMENSIONAL ARRAYS."
0580 PRINT
0590 PRINT "                                    BEGIN TEST."
0600 PRINT
0610 DIM D(20)
0620 FOR I=0 TO 20
0625 LET H=I+1
0630 LET D(I)=H
0640 NEXT I
0650 LET D1=0
0660 LET C=0
0670 FOR I=0 TO 20
0675 LET M=D(I)
0680 LET D1=D1+M
0690 LET C=C+1
0700 NEXT I
0710 IF C<>21 THEN 730
0720 IF D1=231 THEN 750
0730 PRINT "          USE OF DIM FOR SINGLE DIMENSIONING, FAILED TEST."
0740 GOTO 770
0750 PRINT "          USE OF DIM FOR SINGLE DIMENSIONING, PASSED TEST."
0760 PRINT
0770 PRINT "                                    END TEST."
0780 PRINT
0790 PRINT
0800 PRINT "          SECTION 39.2.2: USED WITH TWO-DIMENSIONAL ARRAYS."
0810 PRINT
0820 PRINT "                                    BEGIN TEST."
0830 PRINT
0840 DIM N(20,2),P(2,20),R(20,20)
0850 FOR I=0 TO 20
0855 LET I1=I+1
0860 LET N(I,1)=I1
0870 LET N(I,2)=-I
0880 LET N(I,0)=1
0890 FOR J=2 TO 0 STEP -1
0895 LET J1=I*J
0900 LET P(J,I)=J1
0910 NEXT J
0920 NEXT I
0930 FOR K=0 TO 20
0940 FOR L=0 TO 20
0945 LET K1=K+L
0950 LET R(K,L)=K1
0960 NEXT L
0970 NEXT K
0980 LET N1=0
0990 LET P1=0
1000 LET R1=0
1010 LET C=0
```

```
1020 LET T=0
1030 FOR I=0 TO 20
1040 FOR J=0 TO 2
1045 LET F1=N(I,J)
1050 LET N1=N1+F1
1055 LET G1=P(J,I)
1060 LET P1=P1+G1
1070 LET C=C+1
1080 NEXT J
1090 NEXT I
1100 IF C<>63 THEN 1210
1110 IF N1<>42 THEN 1210
1120 IF P1<>630 THEN 1210
1130 FOR K=20 TO 0 STEP -1
1140 FOR L=0 TO 20
1145 LET Q1=R(K,L)
1150 LET R1=R1+Q1
1160 LET T=T+1
1170 NEXT L
1180 NEXT K
1190 IF T<>441 THEN 1210
1200 IF R1=8820 THEN 1230
1210 PRINT "                DIM FOR DOUBLE DIMENSIONING, FAILED TEST."
1220 GOTO 1260
1230 PRINT "                DIM FOR DOUBLE DIMENSIONING, PASSED TEST."
1240 PRINT
1250 PRINT "                                        END TEST."
1260 PRINT
1270 PRINT
1280 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 39

SECTION 39.1: IMPLICIT DIMENSIONING.

SECTION 39.1.1: ONE-DIMENSIONAL ARRAYS.

BEGIN TEST.

IMPLICIT SINGLE DIMENSIONING TEST, PASSED.

END TEST.

SECTION 39.1.2: TWO-DIMENSIONAL ARRAYS.

BEGIN TEST.

IMPLICIT DOUBLE DIMENSIONING TEST, PASSED.

END TEST.

SECTION 39.2: THE DIM-STATEMENT WITHOUT AN OPTION-STATEMENT.

SECTION 39.2.1: USED WITH ONE-DIMENSIONAL ARRAYS.

BEGIN TEST.

USE OF DIM FOR SINGLE DIMENSIONING, PASSED TEST.

END TEST.

SECTION 39.2.2: USED WITH TWO-DIMENSIONAL ARRAYS.

BEGIN TEST.

DIM FOR DOUBLE DIMENSIONING, PASSED TEST.

END TEST.

40.0 EXTENDING IF-THEN CAPABILITIES BY USING ONE-DIMENSIONAL ARRAYS IN THE COMPARISON

Since subscripted variables have already been introduced in previous tests, they are now added to the list of possible numeric expressions that can be compared in an IF-THEN-statement. The objectve of the next two test programs is to extend the IF-THEN-statement capability by using subscripted variables.

This section will concentrate on testing relation operations between single dimensioned arrays, simple variables and constants. The first comparisons made are between elements of the same array. Next, an array element is compared with a variable on the left of the relational operator and then the test is reversed. Finally, an element of the array is compared against a constant. There is a two column output of which the first heading is "Comparisons", and the second heading is "Results of IF-THEN Comparisons".

The first column of output lists the numerical values that are being compared for each of the relations. The next six columns form a table containing the six relation symbols =, <, >, <>, >=, and <=. This table should appear blank (or empty), unless an error was made in a relation comparison, in which case an asterisk should appear in the column of the relational symbol for which the comparison was not evaluated properly. For example, if two numbers are equal, then a < comparison should be correctly evaluated as false and the correct transfer made. Otherwise, an asterisk would appear in the table under the < column.

```
********************
* PROGRAM FILE 40 *
********************
```

```
0010 PRINT "PROGRAM FILE 40"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                              SECTION 40.0"
0100 PRINT
0110 PRINT "                                 IF-THEN"
0120 PRINT
0130 PRINT
0140 PRINT "    ***USING NUMERICAL CONSTANTS, ASSIGNED SIMPLE VARIABLES"
0150 PRINT "                                  AND"
0160 PRINT "             ASSIGNED SUBSCRIPTED VARIABLES, TOGETHER.***"
0170 PRINT
0180 PRINT
0190 PRINT
0200 PRINT "                                BEGIN TEST."
0210 PRINT
```

```
0220 PRINT "     ALL INVALID IF-THEN EVALUATIONS WILL BE DESIGNATED BY"
0230 PRINT "AN ASTERISK IN THE COLUMNS OF THOSE RELATIONAL SYMBOLS FOR"
0240 PRINT "WHICH ERROR(S) OCCURRED FOR THAT COMPARATIVE ROW."
0250 PRINT
0260 PRINT TAB(43);"RESULTS"
0270 PRINT TAB(45);"OF"
0280 PRINT TAB(10);"COMPARISONS";TAB(39);"IF-THEN EVALUATIONS"
0290 PRINT
0300 LET A$="="
0310 LET B$="<"
0320 LET C$=">"
0330 LET D$="<>"
0340 LET E$=">="
0350 LET F$="<="
0360 PRINT TAB(34);A$;TAB(39);B$;TAB(44);C$;TAB(49);D$;TAB(55);E$;
0370 PRINT TAB(61);F$
0380 PRINT
0390 DIM A(5)
0400 LET A1=1
0410 LET A(1)=2
0420 LET A(2)=3
0430 LET A(3)=-2
0440 LET A2=-3
0450 LET A(4)=3
0460 LET A(5)=0
0470 LET F=0
0475 REM    COMPARING AN ARRAY ELEMENT WITH ANOTHER ELEMENT
0477 REM    OF THE SAME ARRAY
0480 IF A(2)=A(4) THEN 610
0490 LET A$="*"
0500 LET F=1
0510 GOTO 620
0520 LET B$="*"
0530 LET F=1
0540 GOTO 690
0550 LET C$="*"
0560 LET F=1
0570 GOTO 710
0580 LET D$="*"
0590 LET F=1
0600 GOTO 790
0610 LET A$=" "
0620 IF A(2)<=A(4) THEN 740
0630 LET F$="*"
0640 LET F=1
0650 GOTO 750
0660 LET E$=" "
0670 IF A(2)<A(4) THEN 520
0680 LET B$=" "
0690 IF A(2)>A(4) THEN 550
0700 LET C$=" "
0710 IF A(2)<>A(4) THEN 580
0720 LET D$=" "
0730 GOTO 790
0740 LET F$=" "
0750 IF A(2)>=A(4) THEN 660
0760 LET E$="*"
```

```
0770 LET F=1
0780 GOTO 670
0790 PRINT TAB(12);"3 TO 3";TAB(34);A$;TAB(39);B$;TAB(44);C$;TAB(49);D$;
0800 PRINT TAB(55);E$;TAB(61);F$
0805 REM    COMPARING AN ARRAY ELEMENT WITH A SIMPLE VARIABLE
0807 REM    ON THE LEFT
0810 IF A1<A(1) THEN 940
0820 LET B$="*"
0830 LET F=1
0840 GOTO 950
0850 LET A$="*"
0860 LET F=1
0870 GOTO 1020
0880 LET C$="*"
0890 LET F=1
0900 GOTO 1040
0910 LET E$="*"
0920 LET F=1
0930 GOTO 1120
0940 LET B$=" "
0950 IF A1<>A(1) THEN 1070
0960 LET D$="*"
0970 LET F=1
0980 GOTO 1080
0990 LET F$=" "
1000 IF A1=A(1) THEN 850
1010 LET A$=" "
1020 IF A1>A(1) THEN 880
1030 LET C$=" "
1040 IF A1>=A(1) THEN 910
1050 LET E$=" "
1060 GOTO 1120
1070 LET D$=" "
1080 IF A1<=A(1) THEN 990
1090 LET F$="*"
1100 LET F=1
1110 GOTO 1000
1120 PRINT TAB(12);"1 TO 2";TAB(34);A$;TAB(39);B$;TAB(44);C$;TAB(49);D$;
1130 PRINT TAB(55);E$;TAB(61);F$
1135 REM    COMPARING AN ARRAY ELEMENT WITH A SIMPLE VARIABLE
1137 REM    ON THE RIGHT
1140 IF A(2)>A2 THEN 1270
1150 LET C$="*"
1160 LET F=1
1170 GOTO 1280
1180 LET A$="*"
1190 LET F=1
1200 GOTO 1350
1210 LET B$="*"
1220 LET F=1
1230 GOTO 1370
1240 LET F$="*"
1250 LET F=1
1260 GOTO 1450
1270 LET C$=" "
1280 IF A(2)>=A2 THEN 1400
1290 LET E$="*"
```

```
1300 LET F=1
1310 GOTO 1410
1320 LET D$=" "
1330 IF A(2)=A2 THEN 1180
1340 LET A$=" "
1350 IF A(2)<A2 THEN 1210
1360 LET B$=" "
1370 IF A(2)<=A2 THEN 1240
1380 LET F$=" "
1390 GOTO 1450
1400 LET E$=" "
1410 IF A(2)<>A2 THEN 1320
1420 LET D$="*"
1430 LET F=1
1440 GOTO 1330
1450 PRINT TAB(12);"3 TO -3";TAB(34);A$;TAB(39);B$;TAB(44);C$;TAB(49);
1460 PRINT D$;TAB(55);E$;TAB(61);F$
1465 REM    COMPARING AN ARRAY ELEMENT WITH A CONSTANT ON THE LEFT
1470 IF 2>=A(5) THEN 1600
1480 LET E$="*"
1490 LET F=1
1500 GOTO 1610
1510 LET A$="*"
1520 LET F=1
1530 GOTO 1680
1540 LET B$="*"
1550 LET F=1
1560 GOTO 1700
1570 LET F$="*"
1580 LET F=1
1590 GOTO 1780
1600 LET E$=" "
1610 IF 2>A(5) THEN 1730
1620 LET C$="*"
1630 LET F=1
1640 GOTO 1740
1650 LET D$=" "
1660 IF 2=A(5) THEN 1510
1670 LET A$=" "
1680 IF 2<A(5) THEN 1540
1690 LET B$=" "
1700 IF 2<=A(5) THEN 1570
1710 LET F$=" "
1720 GOTO 1780
1730 LET C$=" "
1740 IF 2<>A(5) THEN 1650
1750 LET D$="*"
1760 LET F=1
1770 GOTO 1660
1780 PRINT TAB(12);"2 TO 0";TAB(34);A$;TAB(39);B$;TAB(44);C$;TAB(49);D$;
1790 PRINT TAB(55);E$;TAB(61);F$
1795 REM    COMPARING AN ARRAY ELEMENT WITH A CONSTANT ON THE RIGHT
1800 IF A(5)=0 THEN 1930
1810 LET A$="*"
1820 LET F=1
1830 GOTO 1940
1840 LET B$="*"
```

```
1850 LET F=1
1860 GOTO 2010
1870 LET C$="*"
1880 LET F=1
1890 GOTO 2030
1900 LET D$="*"
1910 LET F=1
1920 GOTO 2110
1930 GOTO 2110
1940 IF A(5)>=0 THEN 2060
1950 LET E$="*"
1960 LET F=1
1970 GOTO 2070
1980 LET F$=" "
1990 IF A(5)>0 THEN 1840
2000 LET B$=" "
2010 IF A(5)<0 THEN 1870
2020 LET C$=" "
2030 IF A(5)<>0 THEN 1900
2040 LET D$=" "
2050 GOTO 2110
2060 LET E$=" "
2070 IF A(5)<=0 THEN 1980
2080 LET F$="*"
2090 LET F=1
2100 GOTO 1990
2110 PRINT TAB(12);"0 TO 0";TAB(34);A$;TAB(39);B$;TAB(44);C$;TAB(49);D$;
2120 PRINT TAB(55);E$;TAB(61);F$
2130 PRINT
2140 IF F<>0 THEN 2190
2150 PRINT TAB(31);"NO ASTERISKS"
2160 PRINT TAB(32);"THEREFORE"
2170 PRINT TAB(31);"TEST PASSED."
2180 GOTO 2220
2190 PRINT TAB(30);"SOME ASTERISKS"
2200 PRINT TAB(32);"THEREFORE"
2210 PRINT TAB(31);"TEST FAILED."
2220 PRINT
2230 PRINT "                                    END TEST."
2240 PRINT
2250 PRINT
2260 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 40

SECTION 40.0
IF-THEN

***USING NUMERICAL CONSTANTS, ASSIGNED SIMPLE VARIABLES
AND
ASSIGNED SUBSCRIPTED VARIABLES, TOGETHER.***

BEGIN TEST.

ALL INVALID IF-THEN EVALUATIONS WILL BE DESIGNATED BY AN ASTERISK IN THE COLUMNS OF THOSE RELATIONAL SYMBOLS FOR WHICH ERROR(S) OCCURRED FOR THAT COMPARATIVE ROW.

COMPARISONS	RESULTS OF IF-THEN EVALUATIONS					
	=	<	>	<>	>=	<=
3 TO 3						
1 TO 2						
3 TO -3						
2 TO 0						
0 TO 0						

NO ASTERISKS
THEREFORE
TEST PASSED.

END TEST.

41.0 EXTENDING IF-THEN CAPABILITIES BY USING TWO-DIMENSIONAL ARRAYS IN THE COMPARISON

The program below is nearly parallel in structure to that in section 40.0. The output format is also much the same. This test program, however, exercises the use of a two dimensioned array in the comparison expression. It begins by comparing a two-dimensional array element with itself, then with a constant, thirdly with array elements from another doubly dimensioned array, fourthly with simple variables, and finally with array elements from a singly dimensioned array.

```
********************
* PROGRAM FILE 41 *
********************
```

```
0010 PRINT "PROGRAM FILE 41"
0050 PRINT "                                SECTION 41.0"
0060 PRINT
0070 PRINT "                                   IF-THEN"
0080 PRINT
0090 PRINT
0100 PRINT "    ***USING NUMERICAL CONSTANTS, ASSIGNED SIMPLE VARIABLES"
0110 PRINT "                                    AND"
0120 PRINT "             ASSIGNED SUBSCRIPTED VARIABLES, TOGETHER.***"
0130 PRINT
0140 PRINT
0150 PRINT
0160 PRINT "                                  BEGIN TEST."
0170 PRINT
0180 PRINT "     ALL INVALID IF-THEN EVALUATIONS WILL BE DESIGNATED BY"
0190 PRINT "AN ASTERISK IN THE COLUMNS OF THOSE RELATIONAL SYMBOLS FOR"
0200 PRINT "WHICH ERROR(S) OCCURRED FOR THAT COMPARATIVE ROW."
0210 PRINT
0220 PRINT
0230 PRINT TAB(43);"RESULTS"
0240 PRINT TAB(45);"OF"
0250 PRINT TAB(10);"COMPARISONS";TAB(39);"IF-THEN EVALUATIONS"
0260 PRINT
0270 LET A$="="
0280 LET B$="<"
0290 LET C$=">"
0300 LET D$="<>"
0310 LET E$=">="
0320 LET F$="<="
0330 PRINT TAB(34);A$;TAB(39);B$;TAB(44);C$;TAB(49);D$;TAB(55);E$;
0340 PRINT TAB(61);F$
0350 PRINT
0360 DIM B(5,5), C(5,5)
```

```
0370 LET A1=2
0380 LET B(1,1)=-1
0390 LET B(2,2)=-2
0400 LET B(3,3)=-3
0410 LET B(4,4)=-2
0415 LET C(4,4)=-2
0420 LET A2=3
0430 LET B(5,5)=0
0440 LET F=0
0445 REM    COMPARING TWO-DIMENSIONAL ARRAY ELEMENT WITH ANOTHER
0447 REM    ELEMENT OF THE SAME ARRAY
0450 IF B(2,2)=B(4,4) THEN 580
0460 LET A$="*"
0470 LET F=1
0480 GOTO 590
0490 LET B$="*"
0500 LET F=1
0510 GOTO 660
0520 LET C$="*"
0530 LET F=1
0540 GOTO 680
0550 LET D$="*"
0560 LET F=1
0570 GOTO 760
0580 LET A$=" "
0590 IF B(2,2)>=B(4,4) THEN 710
0600 LET E$="*"
0610 LET F=1
0620 GOTO 720
0630 LET F$=" "
0640 IF B(2,2)<B(4,4) THEN 490
0650 LET B$=" "
0660 IF B(2,2)>B(4,4) THEN 520
0670 LET C$=" "
0680 IF B(2,2)<>B(4,4) THEN 550
0690 LET D$=" "
0700 GOTO 760
0710 LET E$=" "
0720 IF B(2,2)<=B(4,4) THEN 630
0730 LET F$="*"
0740 LET F=1
0750 GOTO 640
0760 PRINT TAB(10);"-2 TO -2";TAB(34);A$;TAB(39);B$;TAB(44);C$;TAB(49);
0770 PRINT D$;TAB(55);E$;TAB(61);F$
0775 REM    COMPARING A CONSTANT WITH A TWO-DIMENSIONAL ARRAY ELEMENT
0780 IF -3<B(1,1) THEN 910
0790 LET B$="*"
0800 LET F=1
0810 GOTO 920
0820 LET A$="*"
0830 LET F=1
0840 GOTO 990
0850 LET C$="*"
0860 LET F=1
0870 GOTO 1010
0880 LET E$="*"
0890 LET F=1
```

```
0900 GOTO 1090
0910 LET B$=" "
0920 IF -3<>B(1,1) THEN 1040
0930 LET D$="*"
0940 LET F=1
0950 GOTO 1050
0960 LET F$=" "
0970 IF -3=B(1,1) THEN 820
0980 LET A$=" "
0990 IF -3>B(1,1) THEN 850
1000 LET C$=" "
1010 IF -3>=B(1,1) THEN 880
1020 LET E$=" "
1030 GOTO 1090
1040 LET D$=" "
1050 IF -3<=B(1,1) THEN 960
1060 LET F$="*"
1070 LET F=1
1080 GOTO 970
1090 PRINT TAB(10);"-3 TO -1";TAB(34);A$;TAB(39);B$;TAB(44);C$;TAB(49);
1100 PRINT D$;TAB(55);E$;TAB(61);F$
1105 REM    COMPARING A TWO-DIMENSIONAL ARRAY ELEMENT WITH AN ELEMENT
1107 REM    OF ANOTHER TWO-DIMENSIONAL ARRAY
1110 IF B(5,5)>B(4,4) THEN 1240
1120 LET C$="*"
1130 LET F=1
1140 GOTO 1250
1150 LET A$="*"
1160 LET F=1
1170 GOTO 1320
1180 LET B$="*"
1190 LET F=1
1200 GOTO 1340
1210 LET F$="*"
1220 LET F=1
1230 GOTO 1420
1240 LET C$=" "
1250 IF B(5,5)<>C(4,4) THEN 1370
1260 LET D$="*"
1270 LET F=1
1280 GOTO 1380
1290 LET E$=" "
1300 IF B(5,5)=C(4,4) THEN 1150
1310 LET A$=" "
1320 IF B(5,5)<C(4,4) THEN 1180
1330 LET B$=" "
1340 IF B(5,5)<=C(4,4) THEN 1210
1350 LET F$=" "
1360 GOTO 1420
1370 LET D$=" "
1380 IF B(5,5)>=C(4,4) THEN 1290
1390 LET E$="*"
1400 LET F=1
1410 GOTO 1300
1420 PRINT TAB(11);"0 TO -2";TAB(34);A$;TAB(39);B$;TAB(44);C$;TAB(49);
1430 PRINT D$;TAB(55);E$;TAB(61);F$
1435 REM    COMPARING A SIMPLE VARIABLE WITH AN ELEMENTARY
```

```
1437 REM    PARENTHESIZED EXPRESSION ENCLOSING AN ARRAY ELEMENT
1440 IF A2=(-B(3,3)) THEN 1570
1450 LET A$="*"
1460 LET F=1
1470 GOTO 1580
1480 LET B$="*"
1490 LET F=1
1500 GOTO 1650
1510 LET C$="*"
1520 LET F=1
1530 GOTO 1670
1540 LET D$="*"
1550 LET F=1
1560 GOTO 1750
1570 LET A$=" "
1580 IF A2>=(-B(3,3)) THEN 1700
1590 LET E$="*"
1600 LET F=1
1610 GOTO 1710
1620 LET F$=" "
1630 IF A2<(-B(3,3)) THEN 1480
1640 LET B$=" "
1650 IF A2>(-B(3,3)) THEN 1510
1660 LET C$=" "
1670 IF A2<>(-B(3,3)) THEN 1540
1680 LET D$=" "
1690 GOTO 1750
1700 LET E$=" "
1710 IF A2<=(-B(3,3)) THEN 1620
1720 LET F$="*"
1730 LET F=1
1740 GOTO 1630
1750 PRINT TAB(11);"3 TO -(-3)";TAB(34);A$;TAB(39);B$;TAB(44);C$;
1760 PRINT TAB(49);D$;TAB(55);E$;TAB(61);F$
1765 REM    COMPARING AN ARRAY ELEMENT WITH A SIMPLE PARENTHESIZED
1767 REM    EXPRESSION INVOLVING A SIMPLE VARIABLE
1770 IF B(2,2)=(-A1) THEN 1900
1780 LET A$="*"
1790 LET F=1
1800 GOTO 1910
1810 LET B$="*"
1820 LET F=1
1830 GOTO 1980
1840 LET C$="*"
1850 LET F=1
1860 GOTO 2000
1870 LET D$="*"
1880 LET F=1
1890 GOTO 2080
1900 LET A$=" "
1910 IF B(2,2)>=(-A1) THEN 2030
1920 LET E$="*"
1930 LET F=1
1940 GOTO 2040
1950 LET F$=" "
1960 IF B(2,2)<(-A1) THEN 1810
1970 LET B$=" "
```

```
1980 IF B(2,2)>(-A1) THEN 1840
1990 LET C$=" "
2000 IF B(2,2)<>(-A1) THEN 1870
2010 LET D$=" "
2020 GOTO 2080
2030 LET E$=" "
2040 IF B(2,2)<=(-A1) THEN 1950
2050 LET F$="*"
2060 LET F=1
2070 GOTO 1960
2080 PRINT TAB(10);"-2 TO -(2)";TAB(34);A$;TAB(39);B$;TAB(44);C$;
2090 PRINT TAB(49);D$;TAB(55);E$;TAB(61);F$
2100 LET A$=" "
2105 LET B$=" "
2110 LET C$=" "
2115 LET D$=" "
2120 LET E$=" "
2125 LET F$=" "
2130 LET F=0
2135 DIM A(5)
2140 LET B(5,5)=-5
2145 LET A(5)=-5
2150 IF B(5,5)=A(5) THEN 2165
2155 LET A$="*"
2160 LET F=1
2165 IF B(5,5)>=A(5) THEN 2180
2170 LET E$="*"
2175 LET F=1
2180 IF B(5,5)<=A(5) THEN 2195
2185 LET F$="*"
2190 LET F=1
2195 IF B(5,5)<A(5) THEN 2215
2200 IF B(5,5)>A(5) THEN 2230
2210 GO TO 2237
2215 LET B$="*"
2220 LET F=1
2225 GO TO 2200
2230 LET C$="*"
2235 LET F=1
2240 PRINT
2245 IF F<>0 THEN 2270
2250 PRINT TAB(31);"NO ASTERISKS"
2255 PRINT TAB(32);"THEREFORE"
2260 PRINT TAB(31);"TEST PASSED."
2265 GOTO 2285
2270 PRINT TAB(30);"SOME ASTERISKS"
2275 PRINT TAB(32);"THEREFORE"
2280 PRINT TAB(31);"TEST FAILED."
2285 PRINT
2290 PRINT "                                   END TEST."
2295 PRINT
2300 PRINT
2305 END
```

```
*****************
* SAMPLE OUTPUT *
*****************
```

PROGRAM FILE 41

SECTION 41.0

IF-THEN

***USING NUMERICAL CONSTANTS, ASSIGNED SIMPLE VARIABLES
AND
ASSIGNED SUBSCRIPTED VARIABLES, TOGETHER.***

BEGIN TEST.

ALL INVALID IF-THEN EVALUATIONS WILL BE DESIGNATED BY AN ASTERISK IN THE COLUMNS OF THOSE RELATIONAL SYMBOLS FOR WHICH ERROR(S) OCCURRED FOR THAT COMPARATIVE ROW.

COMPARISONS	RESULTS OF IF-THEN EVALUATIONS					
	=	<	>	<>	>=	<=
-2 TO -2						
-3 TO -1						
0 TO -2						
3 TO -(-3)						
-2 TO -(2)						
-5 TO -5						

NO ASTERISKS
THEREFORE
TEST PASSED.

END TEST.

42.0 THE ABS FUNCTION WITH SUBSCRIPTED VARIABLES FOR ARGUMENTS

This test verifies that the absolute value function allows subscripted variables as arguments. In this test, both negative and positive numerical constants are assigned to one and two dimensioned variables. The assigned constants are of NR1, NR2, and NR3 form. The subscripted variables are then used as the arguments of the ABS function.

The output has three columns. The first column is labeled "Value of Argument", the second is labeled "True Evaluation", and the third is labeled "System Evaluation". The first column lists the constants that were assigned to the subscripted variable, the second column lists the implementation output expected, and the third column lists the test system evaluation. If any value in the third column is inaccurate, then an asterisk should appear beside it.

```
********************
* PROGRAM FILE 42 *
********************
```

```
0010 PRINT "PROGRAM FILE 42"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                                SECTION 42.0"
0100 PRINT
0110 PRINT "                              THE ABS FUNCTION."
0120 PRINT
0130 PRINT
0140 PRINT "          *USING NUMERICALLY ASSIGNED SUBSCRIPTED VARIABLES"
0150 PRINT "                                    FOR"
0160 PRINT "                                ARGUMENTS.*"
0170 PRINT
0180 PRINT
0190 PRINT "                                BEGIN TEST."
0200 PRINT
0210 DIM A(6),B(2,3)
0220 LET A(1)=-10
0230 LET B(1,1)=15
0240 LET B(1,2)=-.5
0250 LET A(2)=.125
0260 LET A(3)=-62.5E-3
0270 LET B(1,3)=312.5E-4
0280 LET A(4)=ABS(A(1))
0290 IF A(4)<>10 THEN 320
0300 LET A$=" "
0310 GOTO 330
```

```
0320 LET A$="*"
0330 LET B(2,1)=ABS(B(1,1))
0340 IF B(2,1)<>15 THEN 370
0350 LET B$=" "
0360 GOTO 380
0370 LET B$="*"
0380 LET A(5)=ABS(B(1,2))
0390 IF A(5)<>.5 THEN 420
0400 LET C$=" "
0410 GOTO 430
0420 LET C$="*"
0430 LET B(2,2)=ABS(A(2))
0440 IF B(2,2)<>.125 THEN 470
0450 LET D$=" "
0460 GOTO 480
0470 LET D$="*"
0480 LET A(6)=ABS(A(3))
0490 IF A(6)<>62.5E-3 THEN 520
0500 LET E$=" "
0510 GOTO 530
0520 LET E$="*"
0530 LET B(2,3)=ABS(B(1,3))
0540 IF B(2,3)<>312.5E-4 THEN 570
0550 LET F$=" "
0560 GOTO 580
0570 LET F$="*"
0580 PRINT
0590 PRINT "     EACH EVALUATED FAILURE OF THE ABS FUNCTION WILL BE DE-"
0600 PRINT "NOTED BY AN ASTERISK BEING PRINTED ON THAT COMPARATIVE ROW"
0610 PRINT "OF OUTPUT.  TEST PASSED IF THERE ARE NOT ANY ASTERISKS."
0630 PRINT
0640 PRINT
0650 PRINT "VALUE OF","  TRUE  ","  SYSTEM  "
0660 PRINT "ARGUMENT","EVALUATION","EVALUATION"
0670 PRINT
0680 PRINT "-10 "," 10 ",A(4);A$
0690 PRINT " 15 "," 15 ",B(2,1);B$
0700 PRINT "-.5 "," .5 ",A(5);C$
0710 PRINT " .125 "," .125 ",B(2,2);D$
0720 PRINT "-62.5E-3 "," .0625 ",A(6);E$
0730 PRINT " 312.5E-4 "," .03125 ",B(2,3);F$
0740 PRINT
0750 PRINT "                                   END TEST."
0760 PRINT
0770 PRINT
0780 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 42

SECTION 42.0

THE ABS FUNCTION.

*USING NUMERICALLY ASSIGNED SUBSCRIPTED VARIABLES
FOR
ARGUMENTS.*

BEGIN TEST.

EACH EVALUATED FAILURE OF THE ABS FUNCTION WILL BE DENOTED BY AN ASTERISK BEING PRINTED ON THAT COMPARATIVE ROW OF OUTPUT. TEST PASSED IF THERE ARE NOT ANY ASTERISKS.

VALUE OF ARGUMENT	TRUE EVALUATION	SYSTEM EVALUATION
-10	10	10
15	15	15
-.5	.5	.5
.125	.125	.125
-62.5E-3	.0625	.0625
312.5E-4	.03125	.03125

END TEST.

43.0 USING ELEMENTARY OPERATIONS ON SUBSCRIPTED VARIABLES ASSIGNED SAME TYPE CONSTANTS

The next several tests verify that the implementation will continue to maintain six digits of precision for the operations addition, subtraction, multiplication, division, and involution when subscripted variables are used as terms or factors of numerical expressions. The first test below uses arrays assigned constants of the same type. This isolates any error to that associated with operating on array elements and not to the constants assigned to them.

43.1 Addition

The objective of this test is the same as for section 22.1, except in this case the numerical constants have been assigned to subscripted variables rather than simple variables. There are four different addition exercises performed, one for each of the type constants NR1, NR2, and NR3, and implicit point scaled. Each exercise adds a double- and a single-dimensional array element. This test has the same output format described in section 22.1.

43.2 Subtraction

The objective here is the same as section 22.2, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. The four different subtraction exercises have been constructed so that the output format is similar to that in section 22.2.

```
********************
* PROGRAM FILE 43 *
********************
```

```
0010 PRINT "PROGRAM FILE 43"
0140 PRINT
0150 PRINT
0160 PRINT
0170 PRINT "                          SECTION 43.0"
0180 PRINT
0190 PRINT "                        (NON-MIXED MODES.)"
0200 PRINT
0210 PRINT
0220 PRINT "                            BEGIN TEST."
0230 PRINT
0240 PRINT
0250 PRINT "                          SECTION 43.1"
0260 PRINT
0270 PRINT "                            ++++++++++++"
```

```
0280 PRINT "                                + ADDITION +"
0290 PRINT "                                ++++++++++++"
0300 PRINT
0310 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
0320 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
0330 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
0340 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
0350 PRINT "SIX PLACE ACCURACY."
0360 PRINT
0370 PRINT
0380 PRINT "ASSIGNMENT 1"
0390 PRINT "     +      ","REQUIRED","SUM OF","ABSOLUTE"
0400 PRINT "ASSIGNMENT 2","  SUM    ","SYSTEM"," ERROR  "
0410 PRINT
0420 PRINT
0430 DIM A(4),B(2,2),C(2,4),D(4)
0440 LET A$=" "
0450 LET A(1)=2
0460 LET B(1,1)=-12
0470 LET D(1)=10
0480 LET C(1,1)=A(1)+B(1,1)
0490 LET C(1,2)=C(1,1)+D(1)
0500 IF ABS(C(1,2))<=1E-4 THEN 520
0510 LET A$="*"
0520 PRINT " 2        "
0530 PRINT " +       =","-10 ",C(1,1),C(1,2);A$
0540 PRINT "-12       "
0550 PRINT
0560 LET A$=" "
0570 LET A(2)=10.5
0580 LET B(1,2)=-32.5
0590 LET D(2)=22.0
0600 LET C(1,3)=A(2)+B(1,2)
0610 LET C(1,4)=C(1,3)+D(2)
0620 IF ABS(C(1,4))<=1E-4 THEN 640
0630 LET A$="*"
0640 PRINT " 10.5     "
0650 PRINT "  +      =","-22 ",C(1,3),C(1,4);A$
0660 PRINT "-32.5     "
0670 PRINT
0680 LET A$=" "
0690 LET A(3)=2.5E20
0700 LET B(2,1)=3.5E21
0710 LET D(3)=-3.75E21
0720 LET C(2,1)=A(3)+B(2,1)
0730 LET C(2,2)=C(2,1)+D(3)
0740 IF ABS(C(2,2))<=1E16 THEN 760
0750 LET A$="*"
0760 PRINT " 2.5E20   "
0770 PRINT "   +     ="," 3.75000E21 ",C(2,1),C(2,2);A$
0780 PRINT " 3.5E21   "
0790 PRINT
0800 LET A$=" "
0810 LET A(4)=3E20
0820 LET B(2,2)=4E20
0830 LET D(4)=-7E20
0840 LET C(2,3)=A(4)+B(2,2)
```

```
0850 LET C(2,4)=C(2,3)+D(4)
0860 IF ABS(C(2,4))<=1E15 THEN 880
0870 LET A$="*"
0880 PRINT " 3E20     "
0890 PRINT "  +      =","  7.00000E20 ",C(2,3),C(2,4);A$
0900 PRINT " 4E20     "
0910 PRINT
0920 PRINT "                                    END TEST."
0930 PRINT
0940 PRINT
0950 PRINT
0960 PRINT "                                   BEGIN TEST."
0970 PRINT
0980 PRINT
0990 PRINT "                                  SECTION 43.2"
1000 PRINT
1010 PRINT "                                 ---------------"
1020 PRINT "                                 - SUBTRACTION -"
1030 PRINT "                                 ---------------"
1040 PRINT
1050 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
1060 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
1070 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
1080 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
1090 PRINT "SIX PLACE ACCURACY."
1100 PRINT
1110 PRINT
1120 PRINT "ASSIGNMENT 1","           ","DIFFERENCE"
1130 PRINT "     -      "," REQUIRED ","    OF    ","ABSOLUTE"
1140 PRINT "ASSIGNMENT 2","DIFFERENCE","  SYSTEM  "," ERROR  "
1150 PRINT
1160 PRINT
1170 DIM E(12),F(4,2)
1180 LET A$=" "
1190 LET E(1)=156
1200 LET E(2)=6
1210 LET E(3)=150
1220 LET F(1,1)=E(1)-E(2)
1230 LET F(1,2)=F(1,1)-E(3)
1240 IF ABS(F(1,2))<=1E-3 THEN 1260
1250 LET A$="*"
1260 PRINT " 156       "
1270 PRINT "  -       =","  156 ",F(1,1),F(1,2);A$
1280 PRINT "  6        "
1290 PRINT
1300 LET A$=" "
1310 LET E(4)=2.55
1320 LET E(5)=-12.55
1330 LET E(6)=-15.1
1340 LET F(2,1)=E(5)-E(4)
1350 LET F(2,2)=F(2,1)-E(6)
1360 IF ABS(F(2,2))<=1E-4 THEN 1380
1370 LET A$="*"
1380 PRINT "-12.55     "
1390 PRINT "   -      =","-15.1 ",F(2,1),F(2,2);A$
1400 PRINT "  2.55     "
1410 PRINT
```

```
1420 LET A$=" "
1430 LET E(7)=2.55E20
1440 LET E(8)=-2.55E20
1450 LET E(9)=5.1E20
1460 LET F(3,1)=E(7)-E(8)
1470 LET F(3,2)=F(3,1)-E(9)
1480 IF ABS(F(3,2))<=1E15 THEN 1500
1490 LET A$="*"
1500 PRINT " 2.55E20  "
1510 PRINT "    -    =","  5.10000E20 ",F(3,1),F(3,2);A$
1520 PRINT "-2.55E20  "
1530 PRINT
1540 LET A$=" "
1550 LET E(10)=1E30
1560 LET E(11)=19E30
1570 LET E(12)=18E30
1580 LET F(4,1)=E(11)-E(10)
1590 LET F(4,2)=F(4,1)-E(12)
1600 IF ABS(F(4,2))<=1E26 THEN 1620
1610 LET A$="*"
1620 PRINT " 19E30    "
1630 PRINT "   -     =","  1.80000E31 ",F(4,1),F(4,2);A$
1640 PRINT "  1E30    "
1650 PRINT
1660 PRINT "                                  END TEST."
1670 PRINT
1680 PRINT
1690 END
```

```
*****************
* SAMPLE OUTPUT *
*****************
```

```
PROGRAM FILE 43

                    SECTION 43.0

                (NON-MIXED MODES.)

                     BEGIN TEST.

                   SECTION 43.1

                     ++++++++++++
                     + ADDITION +
```

++++++++++++

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

ASSIGNMENT 1 + ASSIGNMENT 2		REQUIRED SUM	SUM OF SYSTEM	ABSOLUTE ERROR
2 + -12	=	-10	-10	0
10.5 + -32.5	=	-22	-22	0
2.5E20 + 3.5E21	=	3.75000E21	3.75000E+21	0
3E20 + 4E20	=	7.00000E20	7.00000E+20	0

END TEST.

BEGIN TEST.

SECTION 43.2

- SUBTRACTION -

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

ASSIGNMENT 1 - ASSIGNMENT 2		REQUIRED DIFFERENCE	DIFFERENCE OF SYSTEM	ABSOLUTE ERROR
156 - 6	=	156	150	0

```
-12.55
   -      =     -15.1          -15.1          0
  2.55

 2.55E20
   -      =      5.10000E20     5.10000E+20    0
-2.55E20

 19E30
   -      =      1.80000E31     1.80000E+31    0
  1E30

                              END TEST.
```

44.0 USING ELEMENTARY OPERATIONS ON SUBSCRIPTED VARIABLES ASSIGNED SAME TYPE CONSTANTS (CONTINUED)

44.1 Multiplication

This test is similar to section 23.1, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. The four cases exercised in 43.1 and 43.2 are used with a similar output to section 23.1.

44.2 Division

The objective of this test is similar to section 23.2, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. Again, the four separate exercises are used and the output is similar to section 23.2.

44.3 Involution

The objective of this test is similar to test 23.3, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. The four separate exercises are used and the output is similar to section 23.3.

```
**********
* PROGRAM FILE 44 *
**********
```

```
0010 PRINT "PROGRAM FILE 44"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                                   BEGIN TEST."
0100 PRINT
0110 PRINT
0120 PRINT "                                  SECTION 44.1"
0130 PRINT
0140 PRINT "                                XXXXXXXXXXXXXXXXXX"
0150 PRINT "                                X MULTIPLICATION X"
0160 PRINT "                                XXXXXXXXXXXXXXXXXX"
0170 PRINT
0180 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
0190 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
0200 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
0210 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
```

```
0220 PRINT "SIX PLACE ACCURACY."
0230 PRINT
0240 PRINT
0250 PRINT "ASSIGNMENT 1"
0260 PRINT "     X      ","REQUIRED","PRODUCT OF","ABSOLUTE"
0270 PRINT "ASSIGNMENT 2","PRODUCT ","  SYSTEM  "," ERROR  "
0280 PRINT
0290 PRINT
0300 DIM L(4),M(2,6),N(4)
0310 LET A$=" "
0320 LET L(1)=15
0330 LET M(1,1)=20
0340 LET N(1)=300
0350 LET M(1,2)=L(1)*M(1,1)
0360 LET M(1,3)=M(1,2)-N(1)
0370 IF ABS(M(1,3))<=1E-3 THEN 390
0380 LET A$="*"
0390 PRINT " 15       "
0400 PRINT " *       =","  300 ",M(1,2),M(1,3);A$
0410 PRINT " 20       "
0420 PRINT
0430 LET A$=" "
0440 LET L(2)=3.6
0450 LET M(1,4)=4.2
0460 LET N(2)=15.12
0470 LET M(1,5)=L(2)*M(1,4)
0480 LET M(1,6)=M(1,5)-N(2)
0490 IF ABS(M(1,6))<=1E-4 THEN 510
0500 LET A$="*"
0510 PRINT " 3.6      "
0520 PRINT "  *      =","  15.12 ",M(1,5),M(1,6);A$
0530 PRINT " 4.2      "
0540 PRINT
0550 LET A$=" "
0560 LET L(3)=3.6E15
0570 LET M(2,1)=1.2E3
0580 LET N(3)=4.32E18
0590 LET M(2,2)=L(3)*M(2,1)
0600 LET M(2,3)=M(2,2)-N(3)
0610 IF ABS(M(2,3))<=1E13 THEN 630
0620 LET A$=" "
0630 PRINT " 3.6E15   "
0640 PRINT "   *     =","  4.32000E18 ",M(2,2),M(2,3);A$
0650 PRINT " 1.2E3    "
0660 PRINT
0670 LET A$=" "
0680 LET L(4)=3E18
0690 LET M(2,4)=2E-3
0700 LET N(4)=6E15
0710 LET M(2,5)=L(4)*M(2,4)
0720 LET M(2,6)=M(2,5)-N(4)
0730 IF ABS(M(2,6))<=1E10 THEN 750
0740 LET A$="*"
0750 PRINT " 3E18     "
0760 PRINT "  *      =","  6.00000E15 ",M(2,5),M(2,6);A$
0770 PRINT " 2E-3     "
0780 PRINT
```

```
0790 PRINT "                                        END TEST."
0800 PRINT
0810 PRINT
0820 PRINT
0830 PRINT "                                       BEGIN TEST."
0840 PRINT
0850 PRINT
0860 PRINT "                                     SECTION 44.2"
0870 PRINT
0880 PRINT "                                     ////////////"
0890 PRINT "                                     / DIVISION /"
0900 PRINT "                                     ////////////"
0910 PRINT
0920 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
0930 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
0940 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
0950 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
0960 PRINT "SIX PLACE ACCURACY."
0970 PRINT
0980 PRINT
0990 PRINT "ASSIGNMENT 1"
1000 PRINT "     /      ","REQUIRED","QUOTIENT OF","ABSOLUTE"
1010 PRINT "ASSIGNMENT 2","QUOTIENT","  SYSTEM   "," ERROR  "
1020 PRINT
1030 PRINT
1040 DIM X(2,2),Y(12),Z(2,2)
1050 LET A$=" "
1060 LET X(1,1)=15
1070 LET Y(1)=5
1080 LET Z(1,1)=3
1090 LET Y(2)=X(1,1)/Y(1)
1100 LET Y(3)=Y(2)-Z(1,1)
1110 IF ABS(Y(3))<=1E-5 THEN 1130
1120 LET A$="*"
1130 PRINT " 15       "
1140 PRINT " /       =","  3 ",Y(2),Y(3);A$
1150 PRINT " 5        "
1160 PRINT
1170 LET A$=" "
1180 LET X(1,2)=14.2
1190 LET Y(4)=7.1
1200 LET Z(1,2)=2.0
1210 LET Y(5)=X(1,2)/Y(4)
1220 LET Y(6)=Y(5)-Z(1,2)
1230 IF ABS(Y(6))<=1E-5 THEN 1250
1240 LET A$="*"
1250 PRINT " 14.2     "
1260 PRINT "  /      =","  2 ",Y(5),Y(6);A$
1270 PRINT "  7.1     "
1280 PRINT
1290 LET A$=" "
1300 LET X(2,1)=3.5E30
1310 LET Y(7)=7.0E10
1320 LET Z(2,1)=5.0E19
1330 LET Y(8)=X(2,1)/Y(7)
1340 LET Y(9)=Y(8)-Z(2,1)
1350 IF ABS(Y(9))<=1E14 THEN 1370
```

```
1360 LET A$="*"
1370 PRINT " 3.5E30  "
1380 PRINT "   /    =","  5.00000E19 ",Y(8),Y(9);A$
1390 PRINT " 7.0E10  "
1400 PRINT
1410 LET A$=" "
1420 LET X(2,2)=18E20
1430 LET Y(10)=9E-2
1440 LET Z(2,2)=2E22
1450 LET Y(11)=X(2,2)/Y(10)
1460 LET Y(12)=Y(11)-Z(2,2)
1470 IF ABS(Y(12))<=1E17 THEN 1490
1480 LET A$="*"
1490 PRINT " 18E20   "
1500 PRINT "   /    =","  2.00000E22 ",Y(11),Y(12);A$
1510 PRINT "  9E-2   "
1520 PRINT
1530 PRINT "                                  END TEST."
1540 PRINT
1550 PRINT
1560 PRINT
1570 PRINT "                                 BEGIN TEST."
1580 PRINT
1590 PRINT
1600 PRINT "                                SECTION 44.3"
1610 PRINT
1620 PRINT "                               ^^^^^^^^^^^^^^^"
1630 PRINT "                               ^ INVOLUTION ^"
1640 PRINT "                               ^^^^^^^^^^^^^^^"
1650 PRINT
1660 PRINT
1670 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
1680 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
1690 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
1700 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
1710 PRINT "SIX PLACE ACCURACY."
1720 PRINT
1730 PRINT
1740 PRINT "ASSIGNMENT 1"
1750 PRINT "     ^      ","REQUIRED","POWER OF","ABSOLUTE"
1760 PRINT "ASSIGNMENT 2","  POWER  "," SYSTEM "," ERROR  "
1770 PRINT
1780 PRINT
1790 DIM F(6,2),G(4),H(4)
1800 LET A$=" "
1810 LET F(1,1)=-5
1820 LET G(1)=4
1830 LET F(1,2)=625
1840 LET H(1)=F(1,1)^G(1)
1850 LET F(2,1)=H(1)-F(1,2)
1860 IF ABS(F(2,1))<=1E-3 THEN 1880
1870 LET A$="*"
1880 PRINT "-5        "
1890 PRINT "        =","  625 ",H(1),F(2,1);A$
1900 PRINT " 4         "
1910 PRINT
1920 LET A$=" "
```

```
1930 LET F(2,2)=.625
1940 LET G(2)=0.0
1950 LET F(3,1)=1.0
1960 LET H(2)=F(2,2)^G(2)
1970 LET F(3,2)=H(2)-F(3,1)
1980 IF ABS(F(3,2))<=1E-5 THEN 2000
1990 LET A$="*"
2000 PRINT " .625      "
2010 PRINT "         =","  1 ",H(2),F(3,2);A$
2020 PRINT " 0.0       "
2030 PRINT
2040 LET A$=" "
2050 LET F(4,1)=-.005E3
2060 LET G(3)=-200.E-2
2070 LET F(4,2)=4.0E-2
2080 LET H(3)=F(4,1)G(3)
2090 LET F(5,1)=H(3)-F(4,2)
2100 IF ABS(F(5,1))<=1E-7 THEN 2120
2110 LET A$="*"
2120 PRINT "-.005E3    "
2130 PRINT "         =","  .04 ",H(3),F(5,1);A$
2140 PRINT "-200.E-2   "
2150 PRINT
2160 LET A$=" "
2170 LET F(5,2)=400E-2
2180 LET G(4)=5E-1
2190 LET F(6,1)=200E-2
2200 LET H(4)=F(5,2)G(4)
2210 LET F(6,2)=H(4)-F(6,1)
2220 IF ABS(F(6,2))<=1E-5 THEN 2240
2230 LET A$="*"
2240 PRINT " 400E-2    "
2250 PRINT "         =","  2 ",H(4),F(6,2);A$
2260 PRINT "  5E-1     "
2270 PRINT
2280 PRINT "                                    END TEST."
2290 PRINT
2300 PRINT
2310 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

```
PROGRAM FILE 44

                    BEGIN TEST.
```

SECTION 44.1

XXXXXXXXXXXXXXXXX
X MULTIPLICATION X
XXXXXXXXXXXXXXXXX

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

ASSIGNMENT 1 X ASSIGNMENT 2		REQUIRED PRODUCT	PRODUCT OF SYSTEM	ABSOLUTE ERROR
15 * 20	=	300	300	0
3.6 * 4.2	=	15.12	15.12	0
3.6E15 * 1.2E3	=	4.32000E18	4.32000E+18	0
3E18 * 2E-3	=	6.00000E15	6.00000E+15	0

END TEST.

BEGIN TEST.

SECTION 44.2

///////////
/ DIVISION /
///////////

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

ASSIGNMENT 1 / ASSIGNMENT 2		REQUIRED QUOTIENT	QUOTIENT OF SYSTEM	ABSOLUTE ERROR

15 / 5	=	3	3	0
14.2 / 7.1	=	2	2	0
3.5E30 / 7.0E10	=	5.00000E19	5.00000E+19	0
18E20 / 9E-2	=	2.00000E22	2.00000E+22	0

END TEST.

BEGIN TEST.

SECTION 44.3

^^^^^^^^^^^^^^^
^ INVOLUTION ^
^^^^^^^^^^^^^^^

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

ASSIGNMENT 1 ^ ASSIGNMENT 2		REQUIRED POWER	POWER OF SYSTEM	ABSOLUTE ERROR
-5 ^ 4	=	625	625	0
.625 ^ 0.0	=	1	1	0
-.005E3 ^ -200.E-2	=	.04	.04	0
400E-2 ^ 5E-1	=	2	2	0

```
END TEST.
```

45.0 USING ELEMENTARY OPERATIONS ON SUBSCRIPTED VARIABLES ASSIGNED MIXED TYPE CONSTANTS

Addition

The objective of this test is similar to section 24.1, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. There are six separate exercises performed in this routine. First, NR1 and NR2 assigned constants are added. Second, NR1 and NR3 assigned constants are added. Third, NR1 and implicit point scaled constants are assigned, added, then followed by the addition of an NR2 number. Fourth, NR2 and NR3 are combined. Fifth, NR2 and implicit point scaled are combined and, finally, NR3 and implicit point scaled are combined. The output is similar in format to section 24.1.

```
**********
* PROGRAM FILE 45 *
**********
```

```
0010 PRINT "PROGRAM FILE 45"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                                SECTION 45.0"
0100 PRINT
0110 PRINT "                                (MIXED MODES.)"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 PRINT
0190 PRINT "                                 ++++++++++++"
0200 PRINT "                                 + ADDITION +"
0210 PRINT "                                 ++++++++++++"
0220 PRINT
0230 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
0240 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
0250 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
0260 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
0270 PRINT "SIX PLACE ACCURACY."
0280 PRINT
0290 PRINT
0300 PRINT "ASSIGNMENT 1"
0310 PRINT "     +      ","REQUIRED","SUM OF","ABSOLUTE"
0320 PRINT "ASSIGNMENT 2","  SUM    ","SYSTEM"," ERROR  "
0330 PRINT
0340 PRINT
```

```
0350 DIM A(12,2),B(6)
0360 LET A$=" "
0370 LET A(1,1)=12
0380 LET A(1,2)=2.5
0390 LET A(2,1)=-14.5
0400 LET B(1)=A(1,1)+A(1,2)
0410 LET A(2,2)=B(1)+A(2,1)
0420 IF ABS(A(2,2))<=1E-4 THEN 440
0430 LET A$="*"
0440 PRINT " 12           "
0450 PRINT "  +          =","  14.5 ",B(1),A(2,2);A$
0460 PRINT " 2.5          "
0470 PRINT
0480 LET A$=" "
0490 LET A(3,1)=14
0500 LET A(3,2)=12.5E-3
0510 LET A(4,1)=-.140125E2
0520 LET B(2)=A(3,1)+A(3,2)
0530 LET A(4,2)=B(2)+A(4,1)
0540 IF ABS(A(4,2))<=1E-4 THEN 560
0550 LET A$="*"
0560 PRINT "   14         "
0570 PRINT "    +        =","  14.0125 ",B(2),A(4,2);A$
0580 PRINT " 12,5E-3      "
0590 PRINT
0600 LET A$=" "
0610 LET A(5,1)=-9
0620 LET A(5,2)=-15E-4
0630 LET A(6,1)=9.0015
0640 LET B(3)=A(5,1)+A(5,2)
0650 LET A(6,2)=B(3)+A(6,1)
0660 IF ABS(A(6,2))<=1E-5 THEN 680
0670 LET A$="*"
0680 PRINT " -9           "
0690 PRINT "  +          =","-9.0015 ",B(3),A(6,2);A$
0700 PRINT "-15E-4        "
0710 PRINT
0720 LET A$=" "
0730 LET A(7,1)=.625
0740 LET A(7,2)=-.00005E7
0750 LET A(8,1)=499.375
0760 LET B(4)=A(7,1)+A(7,2)
0770 LET A(8,2)=B(4)+A(8,1)
0780 IF ABS(A(8,2))<=1E-3 THEN 800
0790 LET A$="*"
0800 PRINT " .625         "
0810 PRINT "   +         =","-499.375 ",B(4),A(8,2);A$
0820 PRINT "-.0005E7      "
0830 PRINT
0840 LET A$=" "
0850 LET A(9,1)=1234.2
0860 LET A(9,2)=36000E-5
0870 LET A(10,1)=-1234.56
0880 LET B(5)=A(9,1)+A(9,2)
0890 LET A(10,2)=B(5)+A(10,1)
0900 IF ABS(A(10,2))<=1E-2 THEN 910
0910 PRINT " 1234.2       "
```

```
0920 PRINT "     +        =","  1234.56 ",B(5),A(10,2);A$
0930 PRINT "  36000E-5     "
0940 PRINT
0950 LET A$=" "
0960 LET A(11,1)=65.4321E21
0970 LET A(11,2)=12345E17
0980 LET A(12,1)=-6.66666E22
0990 LET B(6)=A(11,1)+A(11,2)
1000 LET A(12,2)=B(6)+A(12,1)
1010 IF ABS(A(12,2))<=1E17 THEN 1030
1020 LET A$="*"
1030 PRINT "  65.4321E21   "
1040 PRINT "      +       =","  6.66666E22 ",B(6),A(12,2);A$
1050 PRINT "  12345E17     "
1060 PRINT
1070 PRINT "                                      END TEST."
1080 PRINT
1090 PRINT
1100 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 45

SECTION 45.0

(MIXED MODES.)

BEGIN TEST.

```
++++++++++++
+ ADDITION +
++++++++++++
```

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

ASSIGNMENT 1

```
    +          REQUIRED       SUM OF          ABSOLUTE
ASSIGNMENT 2     SUM          SYSTEM           ERROR

 12
  +          =  14.5           14.5           0
 2.5

   14
    +        =  14.0125        14.0125        0
 12,5E-3

 -9
  +          = -9.0015        -9.0015         0
-15E-4

 .625
   +         = -499.375       -499.375        0
-.0005E7

 1234.2
    +        =  1234.56        1234.56        0
 36000E-5

 65.4321E21
    +        =  6.66666E22     6.66666E+22    0
 12345E17
```

END TEST.

46.0 USING ELEMENTARY OPERATIONS ON SUBSCRIPTED VARIABLES ASSIGNED MIXED TYPE CONSTANTS (CONTINUED)

The objective of this subtraction test is the same as for section 24.2, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. Six similar exercises to those discussed in section 45.0 are used with the output being similarly formatted to section 24.2.

```
********************
* PROGRAM FILE 46 *
********************
```

```
0010 PRINT "PROGRAM FILE 46"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                                BEGIN TEST."
0100 PRINT
0110 PRINT
0120 PRINT "                               SECTION 46.0"
0130 PRINT
0140 PRINT "                              ---------------"
0150 PRINT "                              - SUBTRACTION -"
0160 PRINT "                              ---------------"
0170 PRINT
0180 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
0185 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
0190 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
0200 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
0210 PRINT "SIX PLACE ACCURACY."
0220 PRINT
0230 PRINT
0240 PRINT "ASSIGNMENT 1","          ","DIFFERENCE"
0250 PRINT "     -      "," REQUIRED ","    OF    ","ABSOLUTE"
0260 PRINT "ASSIGNMENT 2","DIFFERENCE","  SYSTEM  "," ERROR  "
0270 PRINT
0280 PRINT
0290 DIM C(3,6),D(6),E(6)
0300 LET A$=" "
0310 LET C(1,1)=2
0320 LET C(1,2)=.76544
0330 LET C(1,3)=1.23456
0340 LET D(1)=C(1,1)-C(1,2)
0350 LET E(1)=D(1)-C(1,3)
0360 IF ABS(E(1))<=1E-5 THEN 380
0370 LET A$="*"
```

```
0380 PRINT "    2          "
0390 PRINT "    -        =","  1.23456 ",D(1),E(1);A$
0400 PRINT " .76544        "
0410 PRINT
0420 LET A$=" "
0430 LET C(1,4)=15
0440 LET C(1,5)=-.00520E3
0450 LET C(1,6)=.202E2
0460 LET D(2)=C(1,4)-C(1,5)
0470 LET E(2)=D(2)-C(1,6)
0480 IF ABS(E(2))<=1E-4 THEN 500
0490 LET A$="*"
0500 PRINT "   15          "
0510 PRINT "    -        =","  20.2 ",D(2),E(2);A$
0520 PRINT "-.00520E3      "
0530 PRINT
0540 LET A$=" "
0550 LET C(2,1)=-9
0560 LET C(2,2)=87600E-5
0570 LET C(2,3)=-9876E-3
0580 LET D(3)=C(2,1)-C(2,2)
0590 LET E(3)=D(3)-C(2,3)
0600 IF ABS(E(3))<=1E-5 THEN 620
0610 LET A$="*"
0620 PRINT "   -9          "
0630 PRINT "    -        =","-9.876 ",D(3),E(3);A$
0640 PRINT " 87600E-5      "
0650 PRINT
0660 LET A$=" "
0670 LET C(2,4)=8.8
0680 LET C(2,5)=.000231E6
0690 LET C(2,6)=-222.2
0700 LET D(4)=C(2,4)-C(2,5)
0710 LET E(4)=D(4)-C(2,6)
0720 IF ABS(E(4))<=1E-3 THEN 740
0730 LET A$="*"
0740 PRINT "    8.8         "
0750 PRINT "     -        =","-222.2 ",D(4),E(4);A$
0760 PRINT " .000231E6     "
0770 PRINT
0780 LET A$=" "
0790 LET C(3,1)=177.177
0800 LET C(3,2)=540540E-4
0810 LET C(3,3)=123.123
0820 LET D(5)=C(3,1)-C(3,2)
0830 LET E(5)=D(5)-C(3,3)
0840 IF ABS(E(5))<=1E-3 THEN 860
0850 LET A$="*"
0860 PRINT "  177.177      "
0870 PRINT "     -        =","  123.123 ",D(5),E(5);A$
0880 PRINT " 540540E-4     "
0890 PRINT
0900 LET A$=" "
0910 LET C(3,4)=-90.1233E20
0920 LET C(3,5)=-12345E16
0930 LET C(3,6)=-8.88888E21
0940 LET D(6)=C(3,4)-C(3,5)
```

```
0950 LET E(6)=D(6)-C(3,6)
0960 IF ABS(E(6))<=1E16 THEN 970
0970 PRINT "-90.1233E20  "
0980 PRINT "     -      =","-8.88888E21 ",D(6),E(6);A$
0990 PRINT " -12345E16   "
1000 PRINT
1010 PRINT "                                   END TEST."
1020 PRINT
1030 PRINT
1040 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 46

BEGIN TEST.

SECTION 46.0

- SUBTRACTION -

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

ASSIGNMENT 1 - ASSIGNMENT 2		REQUIRED DIFFERENCE	DIFFERENCE OF SYSTEM	ABSOLUTE ERROR
2 - .76544	=	1.23456	1.23456	0
15 - -.00520E3	=	20.2	20.2	0
-9 -	=	-9.876	-9.876	0

```
87600E-5

   8.8
    -       = -222.2          -222.2          0
.000231E6

 177.177
    -       =  123.123         123.123        0
540540E-4

-90.1233E20
    -       = -8.88888E21     -8.88888E+21    0
-12345E16

                                  END TEST.
```

47.0 USING ELEMENTARY OPERATIONS ON SUBSCRIPTED VARIABLES ASSIGNED MIXED TYPE CONSTANTS (CONTINUED)

47.1 Multiplication

The objective of this test is the same as in section 25.1, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. As in sections 45.0 and 46.0, this routine uses six exercises to check the accuracy of simple mixed type multiplication. The output is similar to section 25.1

47.2 Division

The objective of this test is the same as in section 25.2, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. Again, six exercises are used and the output is similar to section 25.2.

47.3 Involution

The objective of this test is the same as in section 25.3, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. Six exercises are used and the output is similar to section 25.3.

```
********************
* PROGRAM FILE 47 *
********************
```

```
0010 PRINT "PROGRAM FILE 47"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                                BEGIN TEST."
0100 PRINT
0110 PRINT
0120 PRINT "                               SECTION 47.1"
0130 PRINT
0140 PRINT "                             XXXXXXXXXXXXXXXXXX"
0150 PRINT "                             X MULTIPLICATION X"
0160 PRINT "                             XXXXXXXXXXXXXXXXXX"
0170 PRINT
0180 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
0190 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
0200 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
```

```
0210 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
0220 PRINT "SIX PLACE ACCURACY."
0230 PRINT
0240 PRINT
0250 PRINT "ASSIGNMENT 1"
0260 PRINT "     *       ","REQUIRED","PRODUCT OF","ABSOLUTE"
0270 PRINT "ASSIGNMENT 2","PRODUCT ","  SYSTEM  "," ERROR  "
0280 PRINT
0290 PRINT
0300 DIM A(6),B(3,2),C(6),D(2,3),E(6)
0310 LET A$=" "
0320 LET A(1)=5
0330 LET B(1,1)=1.25
0340 LET C(1)=6.25
0350 LET D(1,1)=A(1)*B(1,1)
0360 LET E(1)=D(1,1)-C(1)
0370 IF ABS(E(1))<=1E-5 THEN 390
0380 LET A$="*"
0390 PRINT "  5          "
0400 PRINT "  *         =","  6.25 ",D(1,1),E(1);A$
0410 PRINT " 1.25        "
0420 PRINT
0430 LET A$=" "
0440 LET A(2)=21
0450 LET B(1,2)=-.47619E5
0460 LET C(2)=-999999
0470 LET D(1,2)=A(2)*B(1,2)
0480 LET E(2)=D(1,2)-C(2)
0490 IF ABS(E(2))<=1E0 THEN 510
0500 LET A$="*"
0510 PRINT "   21        "
0520 PRINT "    *       =","-999999 ",D(1,2),E(2);A$
0530 PRINT "-.47619E5    "
0540 PRINT
0550 LET A$=" "
0560 LET A(3)=-99
0570 LET B(2,1)=-919100E-2
0580 LET C(3)=909909
0590 LET D(1,3)=A(3)*B(2,1)
0600 LET E(3)=D(1,3)-C(3)
0610 IF ABS(E(3))<=1E0 THEN 630
0620 LET A$="*"
0630 PRINT "    -99       "
0640 PRINT "     *       =","  909909 ",D(1,3),E(3);A$
0650 PRINT "-919100E-2   "
0660 PRINT
0670 LET A$=" "
0680 LET A(4)=.0015
0690 LET B(2,2)=6.25E4
0700 LET C(4)=93.75
0710 LET D(2,1 =A(4)*B(2,2)
0720 LET E(4)=D(2,1)-C(4)
0730 IF ABS(E(4))<=1E-4 THEN 750
0740 LET A$="*"
0750 PRINT " .0015        "
0760 PRINT "   *         =","  93.75 ",D(2,1),E(4);A$
0770 PRINT " 6.25E4       "
```

```
0780 PRINT
0790 LET A$=" "
0800 LET A(5)=1.92
0810 LET B(3,1)=6430E-4
0820 LET C(5)=1.23456
0830 LET D(2,2)=A(5)*B(3,1)
0840 LET E(5)=D(2,2)-C(5)
0850 IF ABS(E(5))<=1E-5 THEN 870
0860 LET A$="*"
0870 PRINT "   1.92      "
0880 PRINT "    *       =","  1.23456 ",D(2,2),E(5);A$
0890 PRINT " 6430E-4     "
0900 PRINT
0910 LET A$=" "
0920 LET A(6)=10.631E27
0930 LET B(3,2)=-72000E5
0940 LET C(6)=-7.65432E37
0950 LET D(2,3)=A(6)*B(3,2)
0960 LET E(6)=D(2,3)-C(6)
0970 IF ABS(E(6))<=1E32 THEN 990
0980 LET A$="*"
0990 PRINT " 10.631E27   "
1000 PRINT "     *      =","-7.65432E37 ",D(2,3),E(6);A$
1010 PRINT "-72000E5     "
1020 PRINT
1030 PRINT "                                   END TEST."
1040 PRINT
1050 PRINT
1060 PRINT
1070 PRINT "                                 BEGIN TEST."
1080 PRINT
1090 PRINT
1100 PRINT "                                SECTION 47.2"
1110 PRINT
1120 PRINT "                                ////////////"
1130 PRINT "                                / DIVISION /"
1140 PRINT "                                ////////////"
1150 PRINT
1160 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
1170 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
1180 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
1190 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
1200 PRINT "SIX PLACE ACCURACY."
1210 PRINT
1220 PRINT
1230 PRINT "ASSIGNMENT 1"
1240 PRINT "     /       ","REQUIRED","QUOTIENT OF","ABSOLUTE"
1250 PRINT "ASSIGNMENT 2","QUOTIENT","   SYSTEM  "," ERROR  "
1260 PRINT
1270 PRINT
1280 DIM L(6),M(2,9),N(6)
1290 LET A$=" "
1300 LET L(1)=625
1310 LET M(1,1)=1.25
1320 LET M(1,2)=500
1330 LET M(1,3)=L(1)/M(1,1)
1340 LET N(1)=M(1,3)-M(1,2)
```

```
1350 IF ABS(N(1))<=1E-3 THEN 1370
1360 LET A$="*"
1370 PRINT " 625        "
1380 PRINT "  /        =","  500 ",M(1,3),N(1);A$
1390 PRINT " 1.25       "
1400 PRINT
1410 LET A$=" "
1420 LET L(2)=84.876E7
1430 LET M(1,4)=-6875
1440 LET M(1,5)=-123456
1450 LET M(1,6)=L(2)/M(1,4)
1460 LET N(2)=M(1,6)-M(1,5)
1470 IF ABS(N(2))<=1E0 THEN 1490
1480 LET A$="*"
1490 PRINT " 84.876E7   "
1500 PRINT "    /     =","-123456 ",M(1,6),N(2);A$

1510 PRINT " -6875      "
1520 PRINT
1530 LET A$=" "
1540 LET L(3)=-198765
1550 LET M(1,7)=-5E-20
1560 LET M(1,8)=39753E20
1570 LET M(1,9)=L(3)/M(1,7)
1580 LET N(3)=M(1,9)-M(1,8)
1590 IF ABS(N(3))<=1E19 THEN 1610
1600 LET A$="*"
1610 PRINT "-198765      "
1620 PRINT "   /       =","  3.9753E24 ",M(1,9),N(3);A$
1630 PRINT "-5E-20       "
1640 PRINT
1650 LET A$=" "
1660 LET L(4)=6.25
1670 LET M(2,1)=2.5E-4
1680 LET M(2,2)=25000.0
1690 LET M(2,3)=L(4)/M(2,1)
1700 LET N(4)=M(2,3)-M(2,2)
1710 IF ABS(N(4))<=1E-1 THEN 1730
1720 LET A$="*"
1730 PRINT "  6.25       "
1740 PRINT "   /       =","  25000 ",M(2,3),N(4);A$
1750 PRINT " 2.5E-4      "
1760 PRINT
1770 LET A$=" "
1780 LET L(5)=.1728
1790 LET M(2,4)=12E12
1800 LET M(2,5)=144000E-19
1810 LET M(2,6)=L(5)/M(2,4)
1820 LET N(5)=M(2,6)-M(2,5)
1830 IF ABS(N(5))<=1E-19 THEN 1850
1840 LET A$="*"
1850 PRINT " .1728       "
1860 PRINT "   /       =","  1.44000E-14 ",M(2,6),N(5);A$
1870 PRINT " 12E12       "
1880 PRINT
1890 LET A$=" "
1900 LET L(6)=-1.25E-10
```

```
1910 LET M(2,7)=625E16
1920 LET M(2,8)=-2.00000E-29
1930 LET M(2,9)=L(6)/M(2,7)
1940 LET N(6)=M(2,9)-M(2,8)
1950 IF ABS(N(6))<=1E-34 THEN 1970
1960 LET A$="*"
1970 PRINT "-1.25E-10  "
1980 PRINT "    /     =","-2.00000E-29 ",M(2,9),N(6);A$
1990 PRINT "  625E16   "
2000 PRINT
2010 PRINT "                                    END TEST."
2020 PRINT
2030 PRINT
2040 PRINT
2050 PRINT "                                  BEGIN TEST."
2060 PRINT
2070 PRINT
2080 PRINT "                                 SECTION 47.3"
2090 PRINT
2100 PRINT "                                ^^^^^^^^^^^^^^^"
2110 PRINT "                                ^ INVOLUTION ^"
2120 PRINT "                                ^^^^^^^^^^^^^^^"
2130 PRINT
2140 PRINT
2150 PRINT
2160 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
2170 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
2180 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
2190 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
2200 PRINT "SIX PLACE ACCURACY."
2210 PRINT
2220 PRINT
2230 PRINT "ASSIGNMENT 1"
2240 PRINT "     ^       ","REQUIRED","POWER OF","ABSOLUTE"
2250 PRINT "ASSIGNMENT 2"," POWER  "," SYSTEM "," ERROR  "
2260 PRINT
2270 PRINT
2280 DIM I(6),J(2,9),K(6)
2290 LET A$=" "
2300 LET I(1)=144
2310 LET J(1,1)=.5
2320 LET J(1,2)=12
2330 LET J(1,3)=I(1)^J(1,1)
2340 LET K(1)=J(1,3)-J(1,2)
2350 IF ABS(K(1))<=1E-4 THEN 2370
2360 LET A$="*"
2370 PRINT " 144        "
2380 PRINT "  ^        =","  12 ",J(1,3),K(1);A$
2390 PRINT "  .5        "
2400 PRINT
2410 LET A$=" "
2420 LET I(2)=5
2430 LET J(1,4)=-.004E3
2440 LET J(1,5)=1.6E-3
2450 LET J(1,6)=I(2)^J(1,4)
2460 LET K(2)=J(1,6)-J(1,5)
2470 IF ABS(K(2))<=1E-8 THEN 2490
```

```
2480 LET A$="*"
2490 PRINT "   5        "
2500 PRINT "   ^       =","  .0016 ",J(1,6),K(2);A$
2510 PRINT "-.004E3     "
2520 PRINT
2530 LET A$=" "
2540 LET I(3)=65536
2550 LET J(1,7)=-625E-4
2560 LET J(1,8)=5E-1
2570 LET J(1,9)=I(3)^J(1,7)
2580 LET K(3)=J(1,9)-J(1,8)
2590 IF ABS(K(3))<=1E-6 THEN 2610
2600 LET A$="*"
2610 PRINT " 65536      "
2620 PRINT "   ^       =","  .5 ",J(1,9),K(3);A$
2630 PRINT "-625E-4     "
2640 PRINT
2650 LET A$=" "
2660 LET I(4)=.03125
2670 LET J(2,1)=-.0002E3
2680 LET J(2,2)=2
2690 LET J(2,3)=I(4)^J(2,1)
2700 LET K(4)=J(2,3)-J(2,2)
2710 IF ABS(K(4))<=1E-5 THEN 2730
2720 LET A$="*"
2730 PRINT " .03125     "
2740 PRINT "   ^       =","  2 ",J(2,3),K(4);A$
2750 PRINT "-.0002E3    "
2760 PRINT
2770 LET A$=" "
2780 LET I(5)=1.2
2790 LET J(2,4)=5000E-3
2800 LET J(2,5)=2.48832
2810 LET J(2,6)=I(5)^J(2,4)
2820 LET K(5)=J(2,6)-J(2,5)
2830 IF ABS(K(5))<=1E-5 THEN 2850
2840 LET A$="*"
2850 PRINT "  1.2       "
2860 PRINT "   ^       =","  2.48832 ",J(2,6),K(5);A$
2870 PRINT " 5000E-3    "
2880 PRINT
2890 LET A$=" "
2900 LET I(6)=1.024E13
2910 LET J(2,7)=-10E-2
2920 LET J(2,8)=5E-2
2930 LET J(2,9)=I(6)^J(2,7)
2940 LET K(6)=J(2,9)-J(2,8)
2950 IF ABS(K(6))<=1E-7 THEN 2970
2960 LET A$="*"
2970 PRINT " 1.024E13   "
2980 PRINT "    ^       =","  .05 ",J(2,9),K(6);A$
2990 PRINT " -10E-2     "
3000 PRINT
3010 PRINT "                                    END TEST."
3020 PRINT
3030 PRINT
3040 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 47

BEGIN TEST.

SECTION 47.1

XXXXXXXXXXXXXXXXX
X MULTIPLICATION X
XXXXXXXXXXXXXXXXX

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

ASSIGNMENT 1 * ASSIGNMENT 2		REQUIRED PRODUCT	PRODUCT OF SYSTEM	ABSOLUTE ERROR
5 * 1.25	=	6.25	6.25	0
21 * -.47619E5	=	-999999	-999999	0
-99 * -919100E-2	=	909909	909909	0
.0015 * 6.25E4	=	93.75	93.75	0
1.92 * 6430E-4	=	1.23456	1.23456	0
10.631E27 * -72000E5	=	-7.65432E37	-7.65432E+37	0

END TEST.

BEGIN TEST.

SECTION 47.2

```
////////////
/ DIVISION /
////////////
```

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

ASSIGNMENT 1 / ASSIGNMENT 2		REQUIRED QUOTIENT	QUOTIENT OF SYSTEM	ABSOLUTE ERROR
625 / 1.25	=	500	500	0
84.876E7 / -6875	=	-123456	-123456	0
-198765 / -5E-20	=	3.9753E24	3.97530E+24	0
6.25 / 2.5E-4	=	25000	25000	0
.1728 / 12E12	=	1.44000E-14	1.44000E-14	0
-1.25E-10 / 625E16	=	-2.00000E-29	-2.00000E-29	0

END TEST.

BEGIN TEST.

SECTION 47.3

INVOLUTION

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

ASSIGNMENT 1 ^ ASSIGNMENT 2		REQUIRED POWER	POWER OF SYSTEM	ABSOLUTE ERROR
144 ^ .5	=	12	12	0
5 ^ -.004E3	=	.0016	.0016	0
65536 ^ -625E-4	=	.5	.5	0
.03125 ^ -.0002E3	=	2	2	0
1.2 ^ 5000E-3	=	2.48832	2.48832	0
1.024E13 ^ -10E-2	=	.05	.05	0

END TEST.

48.0 ADDITION OF MORE THAN TWO TERMS CONTAINING ARRAY ELEMENTS

The objective of this section is to continue the testing of standard conforming numerical expressions. In particular, in this case we exercise the addition of several terms involving array elements. From previous tests we can have confidence in the addition operation on two terms. Here we are extending the capability one more step.

48.1 Using Subscripted Variables

The objective of this test is the same as in section 28.2, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. In this section two expressions are computed, one with five and the other with six terms. They combine single- and double-dimensional arrays. The output is similar to section 28.2.

48.2 Mixing Constants, and Variables

The objective of this test is the same as in section 28.3, except in this test subscripted variables are used along with the numerical constants and simple variables to construct numerical expressions. Two expressions are computed, one with seven and the other with eight terms. Constants, simple variables and arrays are combined to form the expressions.

```
********************
* PROGRAM FILE 48 *
********************
```

```
0010 PRINT "PROGRAM FILE 48"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                              SECTION 48.0"
0100 PRINT
0110 PRINT "                    ADDITION OF MORE THAN TWO TERMS."
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "                                BEGIN TEST."
0160 PRINT
0170 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
0180 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
0190 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
0200 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
0210 PRINT "SIX PLACE ACCURACY."
```

```
0220 PRINT
0230 PRINT
0240 PRINT "NUMBER OF","REQUIRED","SUM OF","ABSOLUTE"
0250 PRINT "  TERMS  ","  SUM    ","SYSTEM"," ERROR  "
0260 PRINT "*************************************************";
0270 PRINT "**********************"
0280 PRINT
0290 PRINT
0300 PRINT "                              SECTION 48.1"
0310 PRINT
0320 PRINT "                                   USING"
0330 PRINT "              NUMERICALLY ASSIGNED SUBSCRIPTED VARIABLES."
0340 PRINT
0350 PRINT
0360 DIM O(6,2),P(10)
0370 LET A$=" "
0380 LET P(1)=2.4
0390 LET O(1,1)=23.05
0400 LET P(2)=230.004
0410 LET O(1,2)=432.1
0420 LET P(3)=300.1
0430 LET O(2,1)=314
0440 LET P(4)=84E-2
0450 LET O(2,2)=.5E-2
0460 LET O(3,1)=-987.654
0470 LET P(5)=P(1)+O(1,1)+P(2)+O(1,2)+P(3)
0480 LET O(3,2)=P(5)+O(3,1)
0490 IF ABS(O(3,2))<=1E-3 THEN 510
0500 LET A$="*"
0510 PRINT "    5    "," 987.654 ",P(5),O(3,2);A$
0520 LET A$=" "
0530 LET O(4,1)=-999.999
0540 LET O(4,2)=O(1,1)+P(2)+O(1,2)+O(2,1)+P(4)+O(2,2)
0550 LET P(6)=O(4,2)+O(4,1)
0560 IF ABS(P(6))<= 1E-3 THEN 580
0570 LET A$="*"
0580 PRINT "    6    "," 999.999 ",O(4,2),P(6);A$
0590 PRINT
0600 PRINT
0610 PRINT "*************************************************";
0620 PRINT "**********************"
0630 PRINT
0640 PRINT
0650 PRINT "                              SECTION 48.2"
0660 PRINT
0670 PRINT "                                   MIXING"
0680 PRINT "NUMERICAL CONS/NUMERICALLY ASSIGNED SIMPLE/SUBSCRIPTED VARS"
0690 PRINT "                                   TOGETHER."
0700 PRINT
0710 PRINT
0720 LET A$=" "
0730 LET Q1=110000
0740 LET P(7)=10.1E7
0750 LET Q2=7.4E8
0760 LET O(5,1)=8E9
0770 LET Q3=1.6E32
0780 LET P(8)=1.2E34
```

```
0790 LET O(5,2)=1.0E36
0800 LET P(9)=Q3+1.3E33+P(8)+100.1E33+100.0E33+O(5,2)+2.1E34
0810 LET O(6,1)=P(9)+(-1.23456E36)
0820 IF ABS(O(6,1))<=1E31 THEN 840
0830 LET A$="*"
0840 PRINT "    7    "," 1.23456E36 ",P(9),O(6,1);A$
0850 LET A$=" "
0860 LET O(6,2)=80000.0+Q1+58E5+P(7)+5.3E7+Q2+1.1E9+O(5,1)
0870 LET P(10)=O(6,2)+(-9.99999E9)
0880 IF ABS(P(10))<=1E4 THEN 900
0890 LET A$="*"
0900 PRINT "    8    "," 9.99999E9 ",O(6,2),P(10);A$
0910 PRINT
0920 PRINT
0930 PRINT "**************************************************";
0940 PRINT "**********************"
0950 PRINT
0960 PRINT "                                   END TEST."
0970 PRINT
0980 END
```

```
*****************
* SAMPLE OUTPUT *
*****************
```

PROGRAM FILE 48

SECTION 48.0

ADDITION OF MORE THAN TWO TERMS.

BEGIN TEST.

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

NUMBER OF TERMS	REQUIRED SUM	SUM OF SYSTEM	ABSOLUTE ERROR

****

SECTION 48.1

USING
NUMERICALLY ASSIGNED SUBSCRIPTED VARIABLES.

5	987.654	987.654	0
6	999.999	999.999	0

****

SECTION 48.2

MIXING
NUMERICAL CONS/NUMERICALLY ASSIGNED SIMPLE/SUBSCRIPTED VARS
TOGETHER.

7	1.23456E36	1.23456E+36	0
8	9.99999E9	9.99999E+9	0

****

END TEST.

49.0 MULTIPLICATION OF MORE THAN TWO TERMS

This section continues the testing of standard conforming numerical expressions.

49.1 Using Subscripted Variables

The objective of this test is the same as in section 29.2, except in this test the numerical constants have been assigned to subscripted variables instead of simple variables. Two expressions of five and six factors each are computed. The output is similar to that of section 29.2.

49.2 Mixing Constants and Variables

The objective of this test is the same as in section 29.3, except in this test subscripted variables along with numerical constants and simple variables are used in the construction of numerical expressions. Two expressions of seven and eight terms respectively are computed. The output is similar to that of section 29.3.

```
********************
* PROGRAM FILE 49 *
********************
```

```
0010 PRINT "PROGRAM FILE 49"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                              SECTION 49.0"
0100 PRINT
0110 PRINT "                  MULTIPLICATION OF MORE THAN TWO TERMS."
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "                                  BEGIN TEST."
0160 PRINT
0170 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
0180 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
0190 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
0200 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF"
0210 PRINT "SIX PLACE ACCURACY."
0220 PRINT
0230 PRINT
0240 PRINT "NUMBER OF","REQUIRED","PRODUCT OF","ABSOLUTE"
0250 PRINT "  TERMS  ","PRODUCT ","  SYSTEM  "," ERROR  "  -
0260 PRINT "*************************************************";
```

```
0270 PRINT "**********************"
0280 PRINT
0290 PRINT
0300 PRINT "                              SECTION 49.1"
0310 PRINT
0320 PRINT "                                   USING"
0330 PRINT "               NUMERICALLY ASSIGNED SUBSCRIPTED VARIABLES."
0340 PRINT
0350 PRINT
0360 DIM Q(13),R(3,4)
0370 LET A$=" "
0380 LET Q(1)=1.5
0390 LET R(1,1)=0.2
0400 LET Q(2)=3.7
0410 LET R(1,2)=2.0
0420 LET Q(3)=2.557
0430 LET R(1,3)=3367
0440 LET Q(4)=3E-11
0450 LET R(1,4)=.14
0460 LET Q(5)=.35E34
0470 LET R(2,1)=2E-11
0480 LET Q(6)=.1
0490 LET R(2,2)=-5.67654
0500 LET Q(7)=-9.89898E13
0510 LET R(2,3)=Q(1)*R(1,1)*Q(2)*R(1,2)*Q(3)
0520 LET Q(8)=R(2,3)+R(2,2)
0530 IF ABS(Q(8))<=1E-5 THEN 550
0540 LET A$="*"
0550 PRINT "    5    ", " 5.67654 ",R(2,3),Q(8);A$
0560 LET A$=" "
0570 LET Q(9)=R(1,3)*Q(4)*R(1,4)*Q(5)*R(2,1)*Q(6)
0580 LET R(2,4)=Q(9)+Q(7)
0590 IF ABS(R(2,4))<=1E8 THEN 610
0600 LET A$="*"
0610 PRINT "    6    "," 9.89898E13 ",Q(9),R(2,4);A$
0620 PRINT
0630 PRINT
0640 PRINT "******************************************************";
0650 PRINT "**********************"
0660 PRINT
0670 PRINT
0680 PRINT "                            SECTION 49.2"
0690 PRINT
0700 PRINT "                                   MIXING"
0710 PRINT "NUMERICAL CONS/NUMERICALLY ASSIGNED SIMPLE/SUBSCRIPTED VARS"
0720 PRINT "                                TOGETHER."
0730 PRINT
0740 PRINT
0750 LET A$=" "
0760 LET S1=1.5E-5
0770 LET Q(10)=0.8E20
0780 LET S2=64.3E8
0790 LET R(3,1)=2.0E-6
0800 LET S3=3
0810 LET Q(11)=6.25
0820 LET S4=10101
0830 LET R(3,2)=1.1
```

```
0840 LET Q(12)=S1*.2E8*Q(10)*.4E-10*S2*.1E-5*R(3,1)
0850 LET R(3,3)=Q(12)+(-1.23456E10)
0860 IF ABS(R(3,3))<=1E5 THEN 880
0870 LET A$="*"
0880 PRINT "    7    "," 1.23456E10 ",Q(12),R(3,3);A$
0890 LET A$=" "
0900 LET R(3,4)=375E10*S3*1.6E-12*Q(11)*4E21*S4*.2E-10*R(3,2)
0910 LET Q(13)=R(3,4)+(-9.99999E16)
0920 IF ABS(Q(13))<=1E11 THEN 940
0930 LET A$="*"
0940 PRINT "    8    "," 9.99999E16 ",R(3,4),Q(13);A$
0950 PRINT
0960 PRINT
0970 PRINT "*****************************************************";
0980 PRINT "**********************"
0990 PRINT
1000 PRINT "                                    END TEST."
1010 PRINT
1020 PRINT
1030 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 49

SECTION 49.0

MULTIPLICATION OF MORE THAN TWO TERMS.

BEGIN TEST.

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF OF SIX PLACE ACCURACY.

NUMBER OF TERMS	REQUIRED PRODUCT	PRODUCT OF SYSTEM	ABSOLUTE ERROR

SECTION 49.1

USING
NUMERICALLY ASSIGNED SUBSCRIPTED VARIABLES.

5	5.67654	5.67654	Ø
6	9.89898E13	9.89898E+13	Ø

****

SECTION 49.2

MIXING
NUMERICAL CONS/NUMERICALLY ASSIGNED SIMPLE/SUBSCRIPTED VARS
TOGETHER.

7	1.23456E1Ø	1.23456E+1Ø	Ø
8	9.99999E16	9.99999E+16	Ø

****

END TEST.

50.0 HIERARCHY OF OPERATORS AND PARENTHESES

The objective of this section is to reconstruct test section 30.0 with subscripted variables. This continues the testing of standard conforming numerical expressions.

50.1 Operators of Equal Priority

The objective of this test is the same as in section 30.1, except in this test the constants have been assigned to subscripted variables. There are five exercises in this test: left-to-right division, left-to-right subtraction, left-to-right involution, left-to-right subtraction and addition, and finally left-to-right division and multiplication. The output is similar to section 30.1.

50.2 Operators of Different Priorities without Parentheses

The objective of this test is the same as in test section 30.2, except in this test all numerical constants have been assigned to subscripted variables. In this part of the program there are three exercises to first test multiplication over addition or subtraction, division over addition or subtraction, and finally to test that involution takes precedence over all other operators. The output is similar to section 30.2.

50.3 Operators of Different Priorities with Parentheses

The objective of this test is the same as in section 30.3, except in this test all numerical constants have been assigned to subscripted variables. In this test, there are two sets of exercises. The first employs simple expressions that use parentheses and finally there is an exercise using more complex expressions. The output is similar to section 30.3.

```
********************
* PROGRAM FILE 50 *
********************
```

```
0010 PRINT "PROGRAM FILE 50"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "      SECTION 50.0: HIERARCHY OF OPERATORS AND PARENTHESES."
0100 PRINT
0110 PRINT
0120 PRINT "     SINCE THIS TEST IS ONLY CONCERNED WITH THE ORDER OF"
0130 PRINT "OPERATIONS, THE SELECTED NUMBERS USED FOR THIS TEST ARE IN"
```

```
0140 PRINT "INTEGER FORM ONLY."
0150 PRINT
0160 PRINT
0170 PRINT
0180 PRINT "                          SECTION 50.1"
0190 PRINT
0200 PRINT "     LEFT TO RIGHT EVALUATION FOR EXPRESSIONS WITH OPERA-"
0210 PRINT "TORS OF EQUAL PRIORITY, USING ASSIGNED SUBSCRIPTED VARIA-"
0220 PRINT "BLES."
0230 PRINT
0240 PRINT
0250 PRINT "*****NOTE: LEFT TO RIGHT EVALUATION FOR EXPRESSIONS WITH"
0260 PRINT "OPERATORS OF ONLY ADDITION OR MULTIPLICATION DOES NOT NEC-"
0270 PRINT "ESSARILY APPLY, THEREFORE, SUCH EXPRESSIONS ARE NOT TESTED"
0280 PRINT "IN THIS TEST.*****"
0290 PRINT
0300 PRINT
0310 PRINT "                                   BEGIN TEST."
0320 PRINT
0330 PRINT TAB(18),"OPERATOR(S)";TAB(40),"EVALUATION"
0340 PRINT TAB(18),"    OF     ";TAB(40),"    OF    "
0350 PRINT TAB(18),"EXPRESSION ";TAB(40),"  SYSTEM  "
0360 PRINT "*************************************************";
0370 PRINT "***********************"
0380 PRINT
0390 PRINT
0400 DIM S(28),T(11,3)
0410 LET A$="PASSED"
0420 LET S(1)=81
0430 LET T(1,1)=9
0440 LET S(2)=3
0450 LET T(1,2)=23
0460 LET S(3)=13
0470 LET T(1,3)=2
0480 LET S(4)=S(1)/T(1,1)/S(2)
0490 LET S(5)=S(4)-3
0500 IF ABS(S(5))<=1E-5 THEN 520
0510 LET A$="FAILED"
0520 PRINT TAB(18),"DIVISION";TAB(40),A$
0530 PRINT
0540 LET A$="PASSED"
0550 LET T(2,1)=T(1,2)-S(3)-T(1,3)
0560 LET T(2,2)=T(2,1)-8
0570 IF ABS(T(2,2))<=1E-5 THEN 590
0580 LET A$="FAILED"
0590 PRINT TAB(18),"SUBTRACTION";TAB(40),A$
0600 PRINT
0610 LET A$="PASSED"
0620 LET T(2,3)=T(1,1)^S(2)T(1,3)
0630 LET T(3,1)=T(2,3)-531441
0640 IF ABS(T(3,1))<=1E0 THEN 660
0650 LET A$="FAILED"
0660 PRINT TAB(18),"EXPONENTIATION";TAB(40),A$
0670 PRINT
0680 LET A$="PASSED"
0690 LET S(6)=T(1,2)-S(3)+T(1,3)
0700 LET S(7)=S(6)-12
```

```
0710 IF ABS(S(7))<=1E-4 THEN 730
0720 LET A$="FAILED"
0730 PRINT TAB(18),"SUBTRACTION"
0740 PRINT TAB(18),"    AND    ";TAB(40),A$
0750 PRINT TAB(18)," ADDITION  "
0760 PRINT
0770 LET A$="PASSED"
0780 LET T(3,2)=S(1)/T(1,1)*S(2)
0790 LET T(3,3)=T(3,2)-27
0800 IF ABS(T(3,3))<=1E-4 THEN 820
0810 LET A$="FAILED"
0820 PRINT TAB(18),"   DIVISION   "
0830 PRINT TAB(18),"     AND      ",TAB(40),A$
0840 PRINT TAB(18),"MULTIPLICATION"
0850 PRINT
0860 PRINT "                                  END TEST."
0870 PRINT
0880 PRINT
0890 PRINT "                          SECTION 50.2"
0900 PRINT
0910 PRINT "     EVALUATION OF THE PRECEDENCE OF OPERATORS FOR EXPRES-"
0920 PRINT "SIONS WHICH CONTAIN OPERATORS OF DIFFERENT PRIORITIES IN"
0930 PRINT "THE ABSENCE OF PARENTHESES, USING ASSIGNED SUBSCRIPTED VAR-"
0940 PRINT "IABLES."
0950 PRINT
0960 PRINT
0970 PRINT "                                  BEGIN TEST."
0980 PRINT
0990 PRINT TAB(21),"PRIORITY";TAB(51),"EVALUATION"
1000 PRINT TAB(24),"OF";TAB(55),"OF"
1010 PRINT TAB(21),"OPERATOR";TAB(53),"SYSTEM"
1020 PRINT "**************************************************";
1030 PRINT "**********"
1040 PRINT
1050 PRINT
1060 LET A$="PASSED"
1070 LET T(4,1)=7
1080 LET S(8)=20
1090 LET T(4,2)=100
1100 LET S(9)=72
1110 LET T(4,3)=6
1120 LET S(10)=10
1130 LET T(5,1)=11
1140 LET S(11)=S(1)+T(1,3)*S(2)-T(4,1)+S(8)
1150 LET S(12)=S(11)-100
1160 IF ABS(S(12))<=1E-3 THEN 1180
1170 LET A$="FAILED"
1180 PRINT TAB(18),"MULTIPLICATION"
1190 PRINT TAB(22),"BEFORE";TAB(53),A$
1200 PRINT TAB(11),"ADDITION OR SUBTRACTION"
1210 PRINT
1220 LET A$="PASSED"
1230 LET T(5,2)=T(4,2)+S(9)/T(1,1)-S(2)-T(4,3)
1240 LET T(5,3)=T(5,2)-99
1250 IF ABS(T(5,3))<=1E-4 THEN 1270
1260 LET A$="FAILED"
1270 PRINT TAB(21),"DIVISION"
```

```
1280 PRINT TAB(22),"BEFORE";TAB(53),A$
1290 PRINT TAB(11),"ADDITION OR SUBTRACTION"
1300 PRINT
1310 LET A$="PASSED"
1320 LET F=0
1330 LET T(6,3)=S(10)+T(4,2)^T(1,3)-T(5,1)
1340 LET T(7,1)=T(6,3)-9999
1350 IF ABS(T(7,1))<=1E-2 THEN 1380
1360 LET F=1
1370 PRINT "EXPONENTIATION BEFORE ADDITION OR SUBTRACTION FAILED."
1380 LET S(13)=T(4,1)*S(2)^T(1,3)/T(1,1)
1385 LET S(14) = S(13) - 7
1390 IF ABS(S(14))<=1E-5 THEN 1420
1400 LET F=1
1410 PRINT "EXPONENTIATION BEFORE MULTIPLICATION OR DIVISION FAILED."
1420 LET T(7,2)=-S(1)^T(1,3)
1430 LET T(7,3)=T(7,2)-(-6561)
1440 IF ABS(T(7,3))<=1E-2 THEN 1470
1450 LET F=1
1460 PRINT "EXPONENTIATION BEFORE UNARY FAILED."
1470 IF F=0 THEN 1490
1480 GOTO 1520
1490 PRINT TAB(18),"EXPONENTIATION"
1500 PRINT TAB(19),"AS FIRST OF";TAB(53),A$
1510 PRINT TAB(20),"OPERATIONS"
1520 PRINT
1530 PRINT "                                  END TEST."
1540 PRINT
1550 PRINT
1560 PRINT "                            SECTION 50.3"
1570 PRINT
1580 PRINT "     EVALUATION OF THE PRECEDENCE OF OPERATIONS FOR THOSE"
1590 PRINT "EXPRESSIONS WHICH CONTAIN OPERATORS OF DIFFERENT PRIORITIES"
1600 PRINT "BUT ARE INFLUENCED BY THE USE OF PARENTHESES, USING AS-"
1610 PRINT "SIGNED SUBSCRIPTED VARIABLES."
1620 PRINT
1630 PRINT
1640 PRINT "                                  BEGIN TEST."
1650 PRINT
1660 PRINT TAB(14),"ALTERATION OF PRIORITY";TAB(51),"EVALUATION"
1670 PRINT TAB(24),"BY";TAB(55),"OF"
1680 PRINT TAB(19),"PARENTHESES";TAB(53),"SYSTEM"
1690 PRINT "*****************************************************";
1700 PRINT "**********************"
1710 PRINT
1720 PRINT
1730 LET F=0
1740 LET S(15)=4
1750 LET T(8,1)=92
1760 LET S(16)=725
1770 LET T(8,2)=274
1780 LET S(17)=1998
1790 LET T(8,3)=27
1800 LET T(9,1)=S(1)/(T(1,1)/S(2))
1810 LET T(9,2)=T(9,1)-27
1820 IF ABS(T(9,2))<=1E-4 THEN 1840
1830 LET F=1
```

```
1840 LET S(18)=T(1,2)-(S(3)-T(1,3))
1850 LET S(19)=S(18)-12
1860 IF ABS(S(19))<=1E-4 THEN 1880
1870 LET F=1
1880 LET S(20)=S(15)^(S(2)T(1,3))
1890 LET S(21)=S(20)-262144
1900 IF ABS(S(21))<=1E0 THEN 1920
1910 LET F=1
1920 LET T(9,3)=T(1,2)-(S(3)+T(1,3))
1930 LET T(10,1)=T(9,3)-8
1940 IF ABS(T(10,1))<=1E-5 THEN 1960
1950 LET F=1
1960 LET S(22)=S(1)/(T(1,1)*S(2))
1970 LET S(23)=S(22)-3
1980 IF ABS(S(23))<=1E-5 THEN 2000
1990 LET F=1
2000 IF F=0 THEN 2030
2010 LET A$="UNSUCCESSFUL"
2020 GOTO 2040
2030 LET A$="SUCCESSFUL"
2040 PRINT TAB(18),"EXPRESSIONS OF"
2050 PRINT TAB(18),"LEFT TO RIGHT";TAB(51),A$
2060 PRINT TAB(20),"EVALUATION"
2070 PRINT
2080 LET F=0
2090 LET T(10,2)=(S(1)+T(1,3))*(S(2)-T(4,1))+S(8)
2100 LET T(10,3)=T(10,2)-(-312)
2110 IF ABS(T(10,3))<=1E-3 THEN 2130
2120 LET F=1
2130 LET S(24)=(T(4,2)+T(8,1))/(T(1,1)-S(2))-T(4,3)
2140 LET S(25)=S(24)-26
2150 IF ABS(S(25))<=1E-4 THEN 2170
2160 LET F=1
2170 LET S(26)=(S(16)+T(8,2))^(S(15)-T(1,3))+S(17)
2180 LET S(27)=S(26)-999999
2190 IF ABS(S(27))<=1E0 THEN 2210
2200 LET F=1
2210 LET T(11,1)=(T(5,1)*S(2))^(T(1,1)/S(2))/T(8,3)
2220 LET T(11,2)=T(11,1)-1331
2230 IF ABS(T(11,2))<=1E-2 THEN 2250
2240 LET F=1
2250 LET T(11,3)=(-S(1))^T(1,3)
2260 LET S(28)=T(11,3)-6561
2270 IF ABS(S(28))<=1E-2 THEN 2290
2280 LET F=1
2290 IF F=0 THEN 2320
2300 LET A$="UNSUCCESSFUL"
2310 GOTO 2330
2320 LET A$="SUCCESSFUL"
2330 PRINT TAB(14),"EXPRESSIONS EVALUATED"
2340 PRINT TAB(18),"BY PRIORITY OF";TAB(51),A$
2350 PRINT TAB(19),"THE OPERATOR"
2360 PRINT
2370 PRINT "                                    END TEST."
2380 PRINT
2390 PRINT
2400 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 50

SECTION 50.0: HIERARCHY OF OPERATORS AND PARENTHESES.

SINCE THIS TEST IS ONLY CONCERNED WITH THE ORDER OF OPERATIONS, THE SELECTED NUMBERS USED FOR THIS TEST ARE IN INTEGER FORM ONLY.

SECTION 50.1

LEFT TO RIGHT EVALUATION FOR EXPRESSIONS WITH OPERATORS OF EQUAL PRIORITY, USING ASSIGNED SUBSCRIPTED VARIABLES.

*****NOTE: LEFT TO RIGHT EVALUATION FOR EXPRESSIONS WITH OPERATORS OF ONLY ADDITION OR MULTIPLICATION DOES NOT NECESSARILY APPLY, THEREFORE, SUCH EXPRESSIONS ARE NOT TESTED IN THIS TEST.*****

BEGIN TEST.

OPERATOR(S) OF EXPRESSION	EVALUATION OF SYSTEM
DIVISION	PASSED
SUBTRACTION	PASSED
EXPONENTIATION	PASSED
SUBTRACTION AND ADDITION	PASSED
DIVISION AND MULTIPLICATION	PASSED

END TEST.

SECTION 50.2

EVALUATION OF THE PRECEDENCE OF OPERATORS FOR EXPRESSIONS WHICH CONTAIN OPERATORS OF DIFFERENT PRIORITIES IN THE ABSENCE OF PARENTHESES, USING ASSIGNED SUBSCRIPTED VARIABLES.

BEGIN TEST.

PRIORITY OF OPERATOR	EVALUATION OF SYSTEM
MULTIPLICATION BEFORE ADDITION OR SUBTRACTION	PASSED
DIVISION BEFORE ADDITION OR SUBTRACTION	PASSED
EXPONENTIATION AS FIRST OF OPERATIONS	PASSED

END TEST.

SECTION 50.3

EVALUATION OF THE PRECEDENCE OF OPERATIONS FOR THOSE EXPRESSIONS WHICH CONTAIN OPERATORS OF DIFFERENT PRIORITIES BUT ARE INFLUENCED BY THE USE OF PARENTHESES, USING ASSIGNED SUBSCRIPTED VARIABLES.

BEGIN TEST.

ALTERATION OF PRIORITY BY PARENTHESES	EVALUATION OF SYSTEM
EXPRESSIONS OF LEFT TO RIGHT	SUCCESSFUL

```
EVALUATION

EXPRESSIONS EVALUATED
BY PRIORITY OF                  SUCCESSFUL
THE OPERATOR

    END TEST.
```

51.0 EVALUATION OF EXPRESSIONS THAT HAVE A VARIETY OF OPERATORS

In this test expressions are formed from numerical constants, numerically assigned simple and assigned subscripted variables. Each expression is characterized by either the absence of parentheses, use of non-nested parentheses, or nested parentheses. The objective of this test is the same as in test section 32.0, except in this test subscripted variables are included along with the numerical constants and simple variables for the construction of the numeric expressions. Three exercises are performed. Each exercise includes the evaluation of eight expressions. In the first exercise, no parentheses are used in order to test left-to-right precedence. In the second exercise, parentheses are used but are not nested in order to test that evaluation within parentheses takes place first and then that left-to-right precedence is observed. Finally, expressions with nested parentheses are evaluated. Output is similar to section 32.0.

```
********************
* PROGRAM FILE 51 *
********************
```

```
0010 PRINT "PROGRAM FILE 51"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 51.0"
0100 PRINT
0110 PRINT "     EVALUATION OF EXPRESSIONS WHICH HAVE A VARIETY OF OP-"
0120 PRINT "ERATORS AND ARE OF ONE OF THREE CATEGORIES:"
0130 PRINT
0140 PRINT "      (1) NO PARENTHESES,"
0150 PRINT "      (2) NON-NESTED PARENTHESES, AND"
0160 PRINT "      (3) NESTED PARENTHESES."
0170 PRINT
0180 PRINT "ALSO, THESE EXPRESSIONS ARE FORMED FROM THE USE OF NUMERI-"
0190 PRINT "CAL CONSTANTS, NUMERICALLY ASSIGNED SIMPLE VARIABLES, AND"
0200 PRINT "NUMERICALLY ASSIGNED SUBSCRIPTED VARIABLES."
0210 PRINT
0220 PRINT
0230 PRINT TAB(18)," CATEGORY ";TAB(36),"EVALUATION"
0240 PRINT TAB(18),"    OF    ";TAB(36),"    OF    "
0250 PRINT TAB(18),"EXPRESSION";TAB(36),"  SYSTEM  "
0260 PRINT "*****************************************************";
0270 PRINT "**********************"
0280 PRINT
0290 PRINT
0300 PRINT "                                BEGIN TEST."
0310 PRINT
```

```
0320 PRINT
0330 DIM Z(3),M(2,8)
0340 LET F=0
0350 LET X=3
0360 LET Y=2
0370 LET Z(1)=5
0380 LET Z(2)=-60
0390 LET Z(3)=92
0400 LET M(1,1)=X+Y*Z(1)-Z(2)/5-Z(2)/5/6+18
0410 LET M(1,2)=M(1,1)-45
0420 IF ABS(M(1,2))<=1E-4 THEN 440
0430 LET F=1
0440 LET M(1,3)=Y^3*4+216/3Y*2+Z(3)-82
0450 LET M(1,4)=M(1,3)-90
0460 IF ABS(M(1,4))<=1E-4 THEN 480
0470 LET F=1
0480 LET M(1,5)=Y*X+Y*Z(2)-Y+170
0490 LET M(1,6)=M(1,5)-54
0500 IF ABS(M(1,6))<=1E-4 THEN 520
0510 LET F=1
0520 LET M(1,7)=Z(2)/Y+105/Z(1)*Y^2+3-24
0530 LET M(1,8)=M(1,7)-33
0540 IF ABS(M(1,8))<=1E-4 THEN 560
0550 LET F=1
0560 LET M(2,1)=Y*Y+Z(2)*Y+167*X+Y-124
0570 LET M(2,2)=M(2,1)-263
0580 IF ABS(M(2,2))<=1E-3 THEN 600
0590 LET F=1
0600 LET M(2,3)=Y*Y+Y*Y+Y*Y+Y*Y+Y-3*Z(1)
0610 LET M(2,4)=M(2,3)-3
0620 IF ABS(M(2,4))<=1E-5 THEN 640
0630 LET F=1
0640 LET M(2,5)=Z(2)+Z(1)+X+Y+Y-X-9
0650 LET M(2,6)=M(2,5)-(-60)
0660 IF ABS(M(2,6))<=1E-4 THEN 680
0670 LET F=1
0680 LET M(2,7)=Z(2)/Z(1)+X+Y*Y^Y-X+10
0690 LET M(2,8)=M(2,7)-6
0700 IF ABS(M(2,8))<=1E-5 THEN 720
0710 LET F=1
0720 IF F<>0 THEN 750
0730 LET A$="PASSED"
0740 GOTO 760
0750 LET A$="FAILED"
0760 LET F=0
0770 LET M(1,1)=(X+Y)*(Z(1)-Z(2))/5-Z(2)/5/6+18
0780 LET M(1,2)=M(1,1)-85
0790 IF ABS(M(1,2))<=1E-4 THEN 810
0800 LET F=1
0810 LET M(1,3)=Y^(3*4)+(216/3)Y*2+(Z(3)-82)
0820 LET M(1,4)=M(1,3)-14474
0830 IF ABS(M(1,4))<=1E-1 THEN 850
0840 LET F=1
0850 LET M(1,5)=Y*(X+Y)*(Z(2)-Y)+170
0860 LET M(1,6)=M(1,5)-(-450)
0870 IF ABS(M(1,6))<=1E-3 THEN 890
0880 LET F=1
```

```
0890 LET M(1,7)=Z(2)/Y+105/Z(1)*Y^(2+3)-24
0900 LET M(1,8)=M(1,7)-618
0910 IF ABS(M(1,8))<=1E-3 THEN 930
0920 LET F=1
0930 LET M(2,1)=Y*(Y+Z(2))*(Y+167)*X+(Y-124)
0940 LET M(2,2)=M(2,1)-(-58934)
0950 IF ABS(E1)<=1E-1 THEN 970
0960 LET F=1
0970 LET M(2,3)=Y*(Y+Y)*Y+(Y*Y+Y)*Y+(Y-3*Z(1))
0980 LET M(2,4)=M(2,3)-15
0990 IF ABS(M(2,4))<=1E-4 THEN 1010
1000 LET F=1
1010 LET M(2,5)=(Z(2)+Z(1)+X+Y+Y)-(X-9)
1020 LET M(2,6)=M(2,5)-(-42)
1030 IF ABS(M(2,6))<=1E-4 THEN 1050
1040 LET F=1
1050 LET M(2,7)=Z(2)/(Z(1)+X)+Y*Y^(Y-X)+10
1060 LET M(2,8)=M(2,7)-3.5
1070 IF ABS(M(2,8))<=1E-4 THEN 1090
1080 LET F=1
1090 IF F<>0 THEN 1120
1100 LET B$="PASSED"
1110 GOTO 1130
1120 LET B$="FAILED"
1130 LET F=0
1140 LET M(1,1)=Y/(184/(Z(3)/(30/(Z(2)/(12/X)))))
1150 LET M(1,2)=M(1,1)-(-.5)
1160 IF ABS(M(1,2))<=1E-6 THEN 1180
1170 LET F=1
1180 LET M(1,3)=Y^3*4+405/(3(Y*2))+Z(3)-82
1190 LET M(1,4)=M(1,3)-47
1200 IF ABS(M(1,4))<=1E-4 THEN 1220
1210 LET F=1
1220 LET M(1,5)=Y*(X+Y*(Z(2)-Y))+170
1230 LET M(1,6)=M(1,5)-(-72)
1240 IF ABS(M(1,6))<=1E-4 THEN 1260
1250 LET F=1
1260 LET M(1,7)=Z(2)/Y+121/(Z(1)*(Y^2+3)-24)
1270 LET M(1,8)=M(1,7)-(-19)
1280 IF ABS(M(1,8))<=1E-4 THEN 1300
1290 LET F=1
1300 LET M(2,1)=Y*(Y+Z(2)*(Y+167*(X+Y)))-124
1310 LET M(2,2)=M(2,1)-(-100560)
1320 IF ABS(M(2,2))<=1E0 THEN 1340
1330 LET F=1
1340 LET M(2,3)=Y*(Y+Y*(Y+Y*(Y+Y*(Y+Y))))-3*Z(1)
1350 LET M(2,4)=M(2,3)-77
1360 IF ABS(M(2,4))<=1E-4 THEN 1380
1370 LET F=1
1380 LET M(2,5)=Z(2)+(Z(1)+(X+(Y+(Y-(X-9)))))
1390 LET M(2,6)=M(2,5)-(-42)
1400 IF ABS(M(2,6))<=1E-4 THEN 1420
1410 LET F=1
1420 LET M(2,7)=Z(2)/(Z(1)+4+(Y*(Y^(Y-X))+10))
1430 LET M(2,8)=M(2,7)-(-3)
1440 IF ABS(M(2,8))<=1E-5 THEN 1460
1450 LET F=1
```

```
1460 IF F<>0 THEN 1490
1470 LET C$="PASSED"
1480 GOTO 1500
1490 LET C$="FAILED"
1500 PRINT TAB(18),"     NO    "
1510 PRINT TAB(18),"PARENTHESES";TAB(38),A$
1520 PRINT
1530 PRINT TAB(18),"PARENTHESES"
1540 PRINT TAB(18),"    BUT    ";TAB(38),B$
1550 PRINT TAB(18),"NON-NESTED "
1560 PRINT
1570 PRINT TAB(18),"   NESTED  "
1580 PRINT TAB(18),"PARENTHESES";TAB(38),C$
1590 PRINT
1600 PRINT "                                    END TEST."
1610 PRINT
1620 PRINT
1630 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 51

SECTION 51.0

EVALUATION OF EXPRESSIONS WHICH HAVE A VARIETY OF OPERATORS AND ARE OF ONE OF THREE CATEGORIES:

(1) NO PARENTHESES,
(2) NON-NESTED PARENTHESES, AND
(3) NESTED PARENTHESES.

ALSO, THESE EXPRESSIONS ARE FORMED FROM THE USE OF NUMERICAL CONSTANTS, NUMERICALLY ASSIGNED SIMPLE VARIABLES, AND NUMERICALLY ASSIGNED SUBSCRIPTED VARIABLES.

```
                          CATEGORY      EVALUATION
                            OF              OF
                         EXPRESSION       SYSTEM
*************************************************************************

                              BEGIN TEST.
```

```
    NO
PARENTHESES     PASSED

PARENTHESES
    BUT         PASSED
NON-NESTED

   NESTED
PARENTHESES     PASSED

    END TEST.
```

52.0 EXCEPTION TEST - ZERO RAISED TO A NEGATIVE POWER

The objective of this test is to verify that the implementation recognizes zero raised to a negative power as an exception with a specified recovery procedure. In this case it means that upon recognition of this error, the implementation should supply machine infinity and continue program execution. The test has a statement at line 290 which allows zero to be raised to a negative power. Some systems may generate diagnostics that refer to this line number. On the other hand, this test prints a statement that informs the user on what he should expect so that the user can for himself determine whether the implementation has performed according to the standard. In particular, the user must look for the machine infinity for his particular system.

```
********************
* PROGRAM FILE 52 *
********************
```

```
0010 PRINT "PROGRAM FILE 52"
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "                              SECTION 52.0"
0140 PRINT
0150 PRINT "                    (ZERO RAISED TO A NEGATIVE POWER.)"
0160 PRINT
0170 PRINT
0180 PRINT
0190 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER "
0200 PRINT "IMPLEMENTATION WILL CONSIDER ZERO BEING RAISED TO A NEGA-"
0210 PRINT "TIVE POWER AS A NONFATAL ERROR; THAT IS, UPON RECOGNITION"
0220 PRINT "OF SUCH AN ERROR, IMPLEMENTATION SHOULD SUPPLY ITS MACHINE"
0230 PRINT "INFINITY AND CONTINUE PROGRAM EXECUTION."
0240 PRINT
0250 PRINT
0260 PRINT
0270 PRINT "                                  BEGIN TEST."
0280 PRINT
0290 LET A=0^-5
0300 PRINT "     IF THE NUMERICAL VALUE, WHICH SHOULD BE PRINTED FOLLOW-"
0310 PRINT "ING THIS MESSAGE, IS RECOGNIZED BY THE USER AS THE MACHINE "
0320 PRINT "INFINITY FOR THE SYSTEM BEING TESTED, THEN IMPLEMENTATION "
0330 PRINT "WILL HAVE PASSED THE TEST."
0340 PRINT
0350 PRINT A
0360 PRINT
0370 PRINT "                                  END TEST."
0380 PRINT
```

```
0390 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 52

SECTION 52.0

(ZERO RAISED TO A NEGATIVE POWER.)

THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER IMPLEMENTATION WILL CONSIDER ZERO BEING RAISED TO A NEGATIVE POWER AS A NONFATAL ERROR; THAT IS, UPON RECOGNITION OF SUCH AN ERROR, IMPLEMENTATION SHOULD SUPPLY ITS MACHINE INFINITY AND CONTINUE PROGRAM EXECUTION.

BEGIN TEST.

? ZERO TO A NEGATIVE POWER IN LINE 290

IF THE NUMERICAL VALUE, WHICH SHOULD BE PRINTED FOLLOWING THIS MESSAGE, IS RECOGNIZED BY THE USER AS THE MACHINE INFINITY FOR THE SYSTEM BEING TESTED, THEN IMPLEMENTATION WILL HAVE PASSED THE TEST.

 1.70141E+38

END TEST.

53.0 EXCEPTION TEST - A NEGATIVE NUMBER RAISED TO A NON-INTEGRAL POWER

The objective of this test is to verify that the implementation recognizes a negative power raised to a non-integral power as an exception with no recovery procedure. This means that the error should be reported and program execution should be suspended pending user-directed restart procedures. In this test, there is a statement at line 270 that has a negative number raised to a non-integral power. A diagnostic should be the only output unless a translator is used that executes line by line. In that case, a fatal diagnostic should appear after the statement BEGIN TEST.

```
********************
* PROGRAM FILE 53 *
********************
```

```
0010 PRINT "PROGRAM FILE 53"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 53.0"
0100 PRINT
0110 PRINT "      (A NEGATIVE NUMBER RAISED TO A NON-INTEGRAL POWER.)"
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0160 PRINT "UPON EVALUATION OF THE OPERATION OF INVOLUTION RESULTING IN"
0170 PRINT "A NEGATIVE NUMBER BEING RAISED TO A NON-INTEGRAL POWER WILL"
0180 PRINT "THE SYSTEM CONSIDER THIS AS A FATAL ERROR.  THAT IS, WILL"
0190 PRINT "THE SYSTEM SUSPEND PROGRAM EXECUTION IN SUCH A WAY THAT US-"
0200 PRINT "ER-DIRECTED RESTART PROCEDURES ARE REQUIRED?  IF THIS PRO-"
0210 PRINT "CEDURE IS TAKEN, THEN THE TEST WILL HAVE PASSED."
0220 PRINT
0230 PRINT
0240 PRINT
0250 PRINT "                                  BEGIN TEST."
0260 PRINT
0270 LET A=(-25)^.5
0280 PRINT A;"HAS BEEN PRINTED, THEREFORE, TEST FAILS WHICH MEANS THE"
0290 PRINT "SYSTEM DID NOT RECOGNIZE PROPERLY A NEGATIVE NUMBER BEING"
0300 PRINT "RAISED TO A NON-INTEGRAL POWER."
0310 PRINT
0320 PRINT "                                    END TEST."
0330 PRINT
0340 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 53

SECTION 53.0

(A NEGATIVE NUMBER RAISED TO A NON-INTEGRAL POWER.)

THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER UPON EVALUATION OF THE OPERATION OF INVOLUTION RESULTING IN A NEGATIVE NUMBER BEING RAISED TO A NON-INTEGRAL POWER WILL THE SYSTEM CONSIDER THIS AS A FATAL ERROR. THAT IS, WILL THE SYSTEM SUSPEND PROGRAM EXECUTION IN SUCH A WAY THAT USER-DIRECTED RESTART PROCEDURES ARE REQUIRED? IF THIS PROCEDURE IS TAKEN, THEN THE TEST WILL HAVE PASSED.

BEGIN TEST.

? ABSOLUTE VALUE RAISED TO POWER IN LINE 270

54.0 SEMANTIC ERROR - SUBSCRIPTED VARIABLE WITH DIFFERENT NUMBERS OF SUBSCRIPTS

The objective of this test is to determine whether the implementation recognizes an occurrence of the same subscripted variable with a different number of subscripts as an error. The routine uses the array A as both a single and double dimensioned array. If an implementation recognizes this error, the output of the test should be a diagnostic that indicates that in lines 340 to 360 there is an array used with different sets of subscripts. However, since this is a semantic rather than a syntactic error some systems may allow one and two - dimensioned arrays to use the same variable name. This test is meant to be informative to the user.

```
********************
* PROGRAM FILE 54 *
********************
```

```
0010 PRINT "PROGRAM FILE 54"
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "                          SECTION 54.0"
0160 PRINT
0170 PRINT "  (THE SAME SUBSCRIPTED VARIABLE WITH DIFFERENT NUMBERS OF"
0180 PRINT "                           SUBSCRIPTS.)"
0190 PRINT
0200 PRINT
0210 PRINT
0220 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0230 PRINT "THE USE OF THE SAME SUBSCRIPTED VARIABLE WITH BOTH ONE SUB-"
0240 PRINT "SCRIPT AND TWO SUBSCRIPTS IN THE SAME PROGRAM IS ALLOWED"
0250 PRINT "BY THE SYSTEM.  IF THE SYSTEM RECOGNIZES THIS AS A FATAL"
0260 PRINT "ERROR, THEN THE TEST WILL HAVE PASSED."
0290 PRINT
0300 PRINT
0310 PRINT
0320 PRINT "                                BEGIN TEST."
0330 PRINT
0340 LET A(1)=2
0350 LET A(5,5)=A(1)+64
0360 PRINT A(5,5);"HAS BEEN PRINTED, THEREFORE, TEST FAILS WHICH MEANS"
0370 PRINT "THE SYSTEM DID NOT RECOGNIZE PROPERLY THE SAME SUBSCRIPTED"
0380 PRINT "VARIABLE OCCURRING WITH DIFFERENT NUMBERS OF SUBSCRIPTS."
0390 PRINT
0400 PRINT "                                 END TEST."
0410 PRINT
0420 END
```

****
* SAMPLE OUTPUT *
****

If this program executes then the test system recognizes the same subscripted variable with one and two - dimensional arrays. However, if a system does not then a possible error diagnostic for this program might be:

```
? DIMENSION ERROR IN LINE 350
```

55.0 EXCEPTION TEST - A SUBSCRIPT IS NOT IN THE RANGE OF THE IMPLICIT DIMENSIONING BOUNDS

The objective of this test is to verify that the implementation will recognize when the values of a subscripted variable are not within the appropriate range. In the present case we test whether the implementation recognizes an exception when a subscript is assigned a value greater than 10 if that array has not been declared in a dimension-statement. There is a statement in which a subscripted variable has a subscript value greater than 10 yet that subscripted variable is not a declared array that might allow for such a subscript value. On output, there should be a fatal diagnostic that might indicate a dimensioning error in line 330.

```
**********
* PROGRAM FILE 55 *
**********
```

```
0010 PRINT "PROGRAM FILE 55"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 55.0"
0100 PRINT
0110 PRINT "(A SUBSCRIPT IS NOT IN THE RANGE OF THE IMPLICIT DIMENSION-"
0120 PRINT "                         ING BOUNDS.)"
0130 PRINT
0140 PRINT
0150 PRINT
0160 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0170 PRINT "THE USE OF A SUBSCRIPT WHICH IS NOT IN THE RANGE OF THE"
0180 PRINT "IMPLICIT DIMENSIONING BOUNDS IS ALLOWED BY THE"
0190 PRINT "SYSTEM.  IF THE SYSTEM RECOGNIZES THIS AS A FATAL ERROR"
0200 PRINT "(THAT IS, SUSPENDING PROGRAM EXECUTION SUCH THAT USER-DI-"
0210 PRINT "RECTED RESTART PROCEDURES ARE REQUIRED), THEN THE TEST WILL"
0220 PRINT "HAVE PASSED."
0230 PRINT
0240 PRINT
0250 PRINT
0270 PRINT
0290 PRINT
0300 PRINT
0310 PRINT "                                   BEGIN TEST."
0320 PRINT
0330 LET A(15)=2
0340 PRINT A(15);"HAS BEEN PRINTED, THEREFORE, TEST FAILS WHICH MEANS"
0350 PRINT "THE SYSTEM DID NOT RECOGNIZE PROPERLY THAT THE SUBSCRIPT"
0360 PRINT "BEING USED WAS OUT OF THE RANGE OF THE IMPLICIT DIMENSION-"
0370 PRINT "ING BOUND."
```

```
0380 PRINT
0390 PRINT "                                   END TEST."
0400 PRINT
0410 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 55

SECTION 55.0

(A SUBSCRIPT IS NOT IN THE RANGE OF THE IMPLICIT DIMENSIONING BOUNDS.)

THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER THE USE OF A SUBSCRIPT WHICH IS NOT IN THE RANGE OF THE IMPLICIT DIMENSIONING BOUNDS IS ALLOWED BY THE SYSTEM. IF THE SYSTEM RECOGNIZES THIS AS A FATAL ERROR (THAT IS, SUSPENDING PROGRAM EXECUTION SUCH THAT USER-DIRECTED RESTART PROCEDURES ARE REQUIRED), THEN THE TEST WILL HAVE PASSED.

BEGIN TEST.

? DIMENSION ERROR IN LINE 330

56.0 EXCEPTION TEST - A SUBSCRIPT IS NOT IN THE RANGE OF AN EXPLICITLY DIMENSIONED VARIABLE

The objective of this test is to verify that the implementation recognizes the assignment of a value to a location greater than the upper bound of the array declaration as an exception. This test has a statement in which an assigned value to the subscript is larger than the upper bound of the array declaration. A fatal diagnostic on output is required. It might indicate that there is a dimension error in line 170.

****
* PROGRAM FILE 56 *
****

```
0010 PRINT "PROGRAM FILE 56"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 56.0"
0100 PRINT "           FATAL ERROR TEST ON DIMENSIONED VARIABLES"
0110 PRINT "                    (THE EXPLICIT CASE.)"
0120 PRINT
0130 PRINT
0140 PRINT "                                BEGIN TEST."
0150 PRINT
0160 DIM A(100)
0170 LET A(101)=64
0180 PRINT A(101);"HAS BEEN PRINTED, THEREFORE, TEST FAILS WHICH MEANS"
0190 PRINT "THE SYSTEM DID NOT RECOGNIZE PROPERLY THAT THE SUBSCRIPT"
0200 PRINT "BEING USED WAS OUT OF THE RANGE OF THE EXPLICIT DIMENSION-"
0210 PRINT "ING BOUND."
0220 PRINT
0230 PRINT "                                  END TEST."
0240 PRINT
0250 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 56

SECTION 56.0
FATAL ERROR TEST ON DIMENSIONED VARIABLES
(THE EXPLICIT CASE.)

BEGIN TEST.

? DIMENSION ERROR IN LINE 170

57.0 ATTEMPTING STRING OVERFLOW BY VARIABLE ASSIGNMENT

The objective of this test is to generate a string overflow to determine whether the implementation will recognize this as an error and provide an appropriate diagnostic. Since a string expression is either a string constant or string variable, two cases must be considered. However, string constants up to the maximum allowed line length have previously been tested. This test, then, will only test string variable assignment. The standard specifies that string variables can have assigned to them, during the execution of a program, a character string from zero to 18 characters. This test attempts to assign strings longer than 18 characters in length. This is an informative test since a portable standard conforming program should only use string variables with assignments of 18 or fewer characters. Implementations that accept the assignment of strings longer than 18 characters are not required to inform the user that this program contains assigned strings, longer than 18 characters. The reader is referred to section 6.4 of BSR X3.60 for the precise specifications.

The test has been constructed with several statements that allow strings of various lengths greater than 18 characters to be assigned to string variables. The lengths of the strings used are 19, 20, 30, 40, 50, and 58 characters in length.

If a diagnostic is generated, then it most likely will terminate the program. Some systems, however, accept long assigned strings. Therefore, since multi-line statements are not permitted, this error test may not generate a diagnostic for systems allowing long strings. This is acceptable since the implementation could not be made to overflow with respect to strings.

```
********************
* PROGRAM FILE 57 *
********************
```

```
0010 PRINT "PROGRAM FILE 57"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "    SECTION 57.0: ERROR TEST ON STRING EXPRESSIONS."
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0140 PRINT "THE ASSIGNING OF MORE THAN 18 CHARACTERS TO A STRING WOULD"
0150 PRINT "BE RECOGNIZED BY THE SYSTEM AS A FATAL ERROR.  THAT IS, UP-"
0160 PRINT "ON SUCH AN ASSIGNMENT, PROGRAM EXECUTION WOULD BE SUSPENDED"
0170 PRINT "PENDING USER-DIRECTED RESTART PROCEDURES.  IF THE SYSTEM"
0180 PRINT "RECOGNIZES SUCH ASSIGNMENTS AS FATAL ERRORS, THEN THE TEST"
```

```
0190 PRINT "PASSES.  HOWEVER, IF IT DOES NOT, THEN THE SYSTEM SATISFIES"
0200 PRINT "MORE THAN WHAT IS REQUIRED BY MINIMAL BASIC."
0210 PRINT
0220 PRINT
0230 PRINT
0240 PRINT "                                BEGIN TEST."
0250 PRINT
0260 LET A$="*********19********"
0270 LET B$="*********20*********"
0280 LET C$="***************30***************"
0290 LET D$="*******************40*******************"
0300 LET E$="************************50************************"
0310 LET F$="****************************58****************************"
0320 PRINT "     IN THE OUTPUT BELOW, THE NUMBERS TOWARD THE CENTER OR"
0330 PRINT "IN THE CENTER OF THE ASTERISKS SIGNIFIES THE LENGTH OF THE"
0340 PRINT "CHARACTER STRINGS ASSOCIATED WITH EACH ASSIGNED STRING VAR-"
0350 PRINT "IABLE."
0360 PRINT
0370 PRINT A$
0380 PRINT B$
0390 PRINT C$
0400 PRINT D$
0410 PRINT E$
0420 PRINT F$
0430 PRINT
0440 PRINT "                                 END TEST."
0450 PRINT
0460 END
```

```
*****************
* SAMPLE OUTPUT *
*****************
```

If a string overflow occurs then a diagnostic message such as:

?STRING TOO LONG IN LINE 260

might appear during execution. Otherwise, the user will get the following printed output:

```
PROGRAM FILE 57

    SECTION 57.0: ERROR TEST ON STRING EXPRESSIONS.
```

THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER THE ASSIGNING OF MORE THAN 18 CHARACTERS TO A STRING WOULD BE RECOGNIZED BY THE SYSTEM AS A FATAL ERROR. THAT IS, UPON SUCH AN ASSIGNMENT, PROGRAM EXECUTION WOULD BE SUSPENDED PENDING USER-DIRECTED RESTART PROCEDURES. IF THE SYSTEM RECOGNIZES SUCH ASSIGNMENTS AS FATAL ERRORS, THEN THE TEST PASSES. HOWEVER, IF IT DOES NOT, THEN THE SYSTEM SATISFIES MORE THAN WHAT IS REQUIRED BY MINIMAL BASIC.

BEGIN TEST.

IN THE OUTPUT BELOW, THE NUMBERS TOWARD THE CENTER OR IN THE CENTER OF THE ASTERISKS SIGNIFIES THE LENGTH OF THE CHARACTER STRINGS ASSOCIATED WITH EACH ASSIGNED STRING VARIABLE.

```
*********19********
*********20*********
**************30**************
********************40********************
**************************50**************************
******************************58******************************
```

END TEST.

58.0 TEST FOR UNDEFINED VARIABLES

At initiation of a program,variables may or may not be assigned a specific value. The objective of this test is to determine which of the following three alternatives for associating implementation-defined initial values with variables is used for the implementation tested. The three alternatives are: (a) all variables receive unknown or arbitrary values (i.e., the implementation takes no explicit action to initialize variables); (b) all numeric variables are assigned the value zero and all string variables the null string; or (c) all variables are recognizeably undefined in the sense that an error will result from any attempt to access the value of a variable before that variable is explicitly assigned a value. The standard recommends that the alternative (c) be adopted in order that a program be much more transportable. The reader should consult section 6.6 of BSR X3.60 for the specifications used here.

This test has numerical expressions that contain undefined variables, that is, a variable which has not been explicitly defined. Which of the above alternatives is practiced by the tested implementation will determine what kind of output there will be. If alternative (a) or (b) is practiced, then the value of the expression will be printed and this value should be followed by a statement to the user regarding the practice of the host implementation. Several runnings of the program would indicate (b) is used if the values 2, 4, 6, 8 are printed consistently. If not then (a) is followed. However, if alternative (c)--which is recommended by the ANSI Minimal BASIC Standard--is practiced, then the output should consist of some form of implementation-defined diagnostics.

```
********************
* PROGRAM FILE 58 *
********************
```

```
0010 PRINT "PROGRAM FILE 58"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "          SECTION 58.0: TEST FOR UNDEFINED VARIABLES."
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0140 PRINT "THE SYSTEM WILL RECOGNIZE AN UNDEFINED VARIABLE AS A FATAL"
0150 PRINT "ERROR, THAT IS, REQUIRING A VALUE TO HAVE BEEN ASSIGNED THE"
0160 PRINT "VARIABLE BEFORE ANY EXPRESSION INVOLVING THAT VARIABLE IS"
0170 PRINT "EXECUTED.  IF THE SYSTEM SHOULD RECOGNIZE THE UNDEFINED"
0180 PRINT "VARIABLE AS A FATAL ERROR (SUSPENDING PROGRAM EXECUTION"
0190 PRINT "PENDING USER-DIRECTED RESTART PROCEDURES), THEN THE TEST"
0200 PRINT "WILL HAVE PASSED; HOWEVER, IF THE SYSTEM DOES NOT MAKE THIS"
```

```
0210 PRINT "RECOGNITION, THEN THE SYSTEM IS SATISFYING MORE THAN IS"
0220 PRINT "REQUIRED BY MINIMAL BASIC."
0230 PRINT
0240 PRINT
0250 PRINT
0260 PRINT "                                    BEGIN TEST."
0270 PRINT
0280 LET A=B+2
0290 PRINT A
0292 LET A=C+4
0293 PRINT A
0294 LET A=D+6
0295 PRINT A
0296 LET A=E+8
0297 PRINT A
0300 PRINT
0305 PRINT "IF THE SEQUENCE 2, 4, 6, 8 HAS BEEN PRINTED ABOVE THEN THE"
0306 PRINT "VARIABLES B, C, D, E WERE INITIALIZED TO 0 (ALTERNATIVE B)."
0307 PRINT "IF ANOTHER SEQUENCE APPEARS THEN ALTERNATIVE A APPLIES."
0310 PRINT
0320 PRINT "                                      END TEST."
0330 PRINT
0340 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

Since the standard suggests that alternative C be adopted, the following diagnostics might appear:

```
? UNINITIALIZED VARIABLE IN LINE 280
? UNINITIALIZED VARIABLE IN LINE 292
? UNINITIALIZED VARIABLE IN LINE 294
? UNINITIALIZED VARIABLE IN LINE 296
```

Otherwise, if alternative B applied the following output might appear:

```
PROGRAM FILE 58


        SECTION 58.0: TEST FOR UNDEFINED VARIABLES.

     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER
```

THE SYSTEM WILL RECOGNIZE AN UNDEFINED VARIABLE AS A FATAL ERROR, THAT IS, REQUIRING A VALUE TO HAVE BEEN ASSIGNED THE VARIABLE BEFORE ANY EXPRESSION INVOLVING THAT VARIABLE IS EXECUTED. IF THE SYSTEM SHOULD RECOGNIZE THE UNDEFINED VARIABLE AS A FATAL ERROR (SUSPENDING PROGRAM EXECUTION PENDING USER-DIRECTED RESTART PROCEDURES), THEN THE TEST WILL HAVE PASSED; HOWEVER, IF THE SYSTEM DOES NOT MAKE THIS RECOGNITION, THEN THE SYSTEM IS SATISFYING MORE THAN IS REQUIRED BY MINIMAL BASIC.

BEGIN TEST.

```
 2
 4
 6
 8
```

IF THE SEQUENCE 2, 4, 6, 8 HAS BEEN PRINTED ABOVE THEN THE VARIABLES B, C, D, F WERE INITIALIZED TO Ø (ALTERNATIVE B). IF ANOTHER SEQUENCE APPEARS THEN ALTERNATIVE A APPLIES.

END TEST.

59.0 EXCEPTION TEST - ON DIVISION BY ZERO

The objective of this test is to verify that the implementation recognizes a numerical expression involving division by zero as an exception with a recovery procedure. When the implementation recognizes this situation, it must supply the machine infinity with the sign of the numerator and continue program execution. The reader is referred to section 7.5 of BSR X3.60 for the specifications.

59.1 Positive Numerator

The objective of this test is to determine, in the event that the implementation does recognize division by zero as a recoverable error, that it will also recognize the sign of the numerator (in this case positive) and assign it to its machine infinity. This test has an expression at line 290 which, when evaluated, will involve division by zero. The numerator for the expression is positive. On output, this test requires that the implementation-supplied machine infinity printed and, preceding this value, there should be a message informing the user to look for the positive case of the machine infinity.

59.2 Negative Numerator

The objective of this test is the same as the stated objective for section 59.1, except that this test uses a negative numerator. The expression in this case is at line 470.

****
* PROGRAM FILE 59 *
****

```
0010 PRINT "PROGRAM FILE 59"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "  SECTION 59.0: NON-FATAL ERROR TEST FOR DIVISION BY ZERO."
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "     THE OBJECTIVE OF THIS TEST IS TO DETEMINE WHETHER THE"
0140 PRINT "EVALUATION OF THE SYSTEM ON A EXPRESSION WHICH RESULTED IN"
0150 PRINT "DIVISION BY ZERO WILL CAUSE THE SYSTEM TO SUPPLY ITS"
0160 PRINT "MACHINE INFINITY WITH THE SIGN OF THE NUMERATOR AND A"
0170 PRINT "CONTINUATION OF PROGRAM."
0180 PRINT
0190 PRINT
0200 PRINT
0210 PRINT "                                SECTION 59.1"
```

```
0220 PRINT
0230 PRINT "                              (POSITIVE NUMERATOR.)"
0240 PRINT
0250 PRINT
0260 PRINT "                                   BEGIN TEST."
0270 PRINT
0280 LET A=2
0290 LET X=32/(A-2)
0300 PRINT "     IF THE NUMBER PRINTED AFTER THIS STATEMENT IS POSITIVE"
0310 PRINT "AND THE MACHINE INFINITY  FOR THE SYSTEM, THEN THE TEST"
0320 PRINT "WILL HAVE PASSED."
0330 PRINT X
0340 PRINT
0350 PRINT "                                     END TEST."
0360 PRINT
0370 PRINT
0380 PRINT
0390 PRINT "                                  SECTION 59.2"
0400 PRINT
0410 PRINT "                              (NEGATIVE NUMERATOR.)"
0420 PRINT
0430 PRINT
0440 PRINT "                                   BEGIN TEST."
0450 PRINT
0460 LET A=64
0470 LET X=(-32)/(A-64)
0480 PRINT "     IF THE NUMBER PRINTED AFTER THIS STATEMENT IS NEGATIVE"
0490 PRINT "AND IS THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST"
0500 PRINT "WILL HAVE PASSED."
0510 PRINT X
0520 PRINT
0530 PRINT "                                     END TEST."
0540 PRINT
0550 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

```
PROGRAM FILE 59


  SECTION 59.0: NON-FATAL ERROR TEST FOR DIVISION BY ZERO.


     THE OBJECTIVE OF THIS TEST IS TO DETEMINE WHETHER THE
EVALUATION OF THE SYSTEM ON A EXPRESSION WHICH RESULTED IN
DIVISION BY ZERO WILL CAUSE THE SYSTEM TO SUPPLY ITS
```

MACHINE INFINITY WITH THE SIGN OF THE NUMERATOR AND A CONTINUATION OF PROGRAM.

SECTION 59.1

(POSITIVE NUMERATOR.)

BEGIN TEST.

?DIVISION BY ZERO IN LINE 290

IF THE NUMBER PRINTED AFTER THIS STATEMENT IS POSITIVE AND THE MACHINE INFINITY FOR THE SYSTEM, THEN THE TEST WILL HAVE PASSED.

1.70141E+38

END TEST.

SECTION 59.2

(NEGATIVE NUMERATOR.)

BEGIN TEST.

?DIVISION BY ZERO IN LINE 470

IF THE NUMBER PRINTED AFTER THIS STATEMENT IS NEGATIVE AND IS THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST WILL HAVE PASSED.

-1.70141E+38

END TEST.

60.0 EXCEPTION TEST - ON EXPRESSION EVALUATION RESULTING IN OVERFLOW

The objective of this test is to verify that the implementation, when attempting to evaluate an expression causing an overflow, will recognize a recoverable exception. Following this overflow , the implementation should supply machine infinity with the algebraically correct sign. The reader is referred to section 7.5 of BSR X3.60 for the specifications.

60.1 Positive Machine Infinity

In this part of the test, we raise 999999 to the 99999 power, which, for all practical purposes, on existing computer systems, is sufficiently large to cause overflow. All of the numbers are kept within 6 digits so that roundoff is not a factor. This test requires that the implementation-supplied machine infinity be printed. Preceding the output of the machine infinity, there should be a printed message informing the user what to look for, in order to verify that implementation passed or failed the test. In this case, the user is instructed to look for positive machine infinity. On the test system used machine infinity was 1.70141E+38.

60.2 Negative Machine Infinity

Two cases are considered here. First, the negative number -999999 is raised to the odd power 99999 then, secondly, it is raised to the even power 88888. In the first case, the implementation should return machine infinity with a negative sign and in the second case, with a positive sign. The implementation supplied machine infinity is printed preceded by the appropriate sign for each case. The user is informed by message what to look for in terms of sign.

```
********************
* PROGRAM FILE 60 *
********************
```

```
0010 PRINT "PROGRAM FILE 60"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "  SECTION 60.0: NON-FATAL ERROR TEST FOR EXPRESSION EVALU-"
0100 PRINT "              ATION WHICH RESULTS IN OVERFLOW."
0110 PRINT
0120 PRINT
0130 PRINT
0140 PRINT "     THE OBJECTIVE OF THIS TEST IS TO DETERMINE WHETHER UP-"
0150 PRINT "ON THE EVALUATION OF AN EXPRESSION WHICH CAUSES OVERFLOW"
0160 PRINT "THE SYSTEM WILL SUPPLY ITS MACHINE INFINITY, WHICH SHOULD BE"
0170 PRINT "ACCOMPANIED BY THE CORRECT ALGEBRAIC SIGN, AND IF PROGRAM"
```

```
0180 PRINT "EXECUTION CONTINUES."
0190 PRINT
0200 PRINT
0210 PRINT
0220 PRINT "                                    SECTION 60.1"
0230 PRINT
0240 PRINT "                           (MACHINE INFINITY, POSITIVE.)"
0250 PRINT
0260 PRINT
0270 PRINT "                                     BEGIN TEST."
0280 PRINT
0290 LET A=99999
0300 LET X=999999^A
0310 PRINT "     IF THE NUMBER PRINTED AFTER THIS STATEMENT IS POSITIVE"
0320 PRINT "AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST"
0330 PRINT "WILL HAVE PASSED."
0340 PRINT X
0350 PRINT
0360 PRINT "                                       END TEST."
0370 PRINT
0380 PRINT
0390 PRINT
0400 PRINT "                                    SECTION 60.2"
0410 PRINT
0420 PRINT "                           (MACHINE INFINITY, NEGATIVE.)"
0430 PRINT
0440 PRINT
0450 PRINT "                                     BEGIN TEST."
0460 PRINT
0470 PRINT
0480 LET A=99999
0490 LET X=(-999999)^A
0500 PRINT "     IF THE NUMBER PRINTED AFTER THIS STATEMENT IS NEGATIVE"
0510 PRINT "AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST"
0520 PRINT "WILL HAVE PASSED."
0530 PRINT X
0540 PRINT
0542 LET A=88888
0543 LET X=(-99999)^A*(-1)*(-5)+2
0545 PRINT "     IF THE NUMBER PRINTED AFTER THIS STATEMENT IS POSITIVE"
0546 PRINT "AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST"
0547 PRINT "WILL HAVE PASSED."
0549 PRINT X
0550 PRINT "                                       END TEST."
0560 PRINT
0570 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 60

SECTION 60.0: NON-FATAL ERROR TEST FOR EXPRESSION EVALUATION WHICH RESULTS IN OVERFLOW.

THE OBJECTIVE OF THIS TEST IS TO DETERMINE WHETHER UPON THE EVALUATION OF AN EXPRESSION WHICH CAUSES OVERFLOW THE SYSTEM WILL SUPPLY ITS MACHINE INFINITY, WHICH SHOULD BE ACCOMPANIED BY THE CORRECT ALGEBRAIC SIGN, AND IF PROGRAM EXECUTION CONTINUES.

SECTION 60.1

(MACHINE INFINITY, POSITIVE.)

BEGIN TEST.

?OVERFLOW IN LINE 300

IF THE NUMBER PRINTED AFTER THIS STATEMENT IS POSITIVE AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST WILL HAVE PASSED.

1.70141E+38

END TEST.

SECTION 60.2

(MACHINE INFINITY, NEGATIVE.)

BEGIN TEST.

?OVERFLOW IN LINE 490

IF THE NUMBER PRINTED AFTER THIS STATEMENT IS NEGATIVE AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST WILL HAVE PASSED.

-1.70141E+38

? OVERFLOW IN LINE 543

IF THE NUMBER PRINTED AFTER THIS STATEMENT IS POSITIVE AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST WILL HAVE PASSED.

1.70141E+38

END TEST.

61.0 SEMANTIC TEST - ON THE MAGNITUDE OF A NONZERO NUMERIC CONSTANT THAT IS TOO SMALL

The objective of this test is to verify that the implementation will recognize a numerical constant, with a magnitude outside the implementation-defined range, as a diagnosable error. Since numeric constants are expressions, their errors are handled in the same manner. If the magnitude of the constant is too small then the implementation should supply 0 and continue. The ANSI Minimal BASIC standard does not require a diagnostic message in an underflow of this kind. If the magnitude is too large then the implementation should supply machine infinity with the appropriate sign. In the case of an overflow a diagnostic message is required. The reader is referred to section 7.4 of BSR X3.60 for the specifications.

This test will determine whether the implementation will supply a value of zero for an extremely small value which most present implementations cannot represent. For any value so close to zero that it is outside of the implementation-defined range, a value of zero should be supplied by the implementation and program execution continued.

This test uses a numerical constant, 10.0E-99999, which is too small to be represented on most present day machines. The constant is assigned on line 310 and diagnostics might refer to this line. On encountering this number, a processor should assign 0 to A and continue.

```
********************
* PROGRAM FILE 61 *
********************
```

```
0010 PRINT "PROGRAM FILE 61"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "  SECTION 61.0: SEMANTIC TEST ON THE MAGNITUDE OF A"
0100 PRINT "                NONZERO NUMERIC CONSTANT."
0110 PRINT "              (THE MAGNITUDE IS TOO SMALL)"
0120 PRINT
0130 PRINT
0140 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0150 PRINT "UPON THE ASSIGNMENT OF A NONZERO CONSTANT WHICH IS TOO"
0160 PRINT "SMALL FOR THE IMPLEMENTATION A ZERO WILL"
0170 PRINT "BE SUPPLIED."
0200 PRINT
0210 PRINT
0220 PRINT
0280 PRINT "                                BEGIN TEST."
0290 PRINT
```

```
0310 LET A=10.0E-99999
0320 PRINT "     IF THE NUMBER PRINTED AFTER THIS STATEMENT IS ZERO,"
0330 PRINT "THEN THE TEST WILL HAVE PASSED."
0340 PRINT A
0350 PRINT
0360 PRINT "                                   END TEST."
0370 PRINT
0380 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 61

SECTION 61.0: SEMANTIC TEST ON THE MAGNITUDE OF A
NONZERO NUMERIC CONSTANT.
(THE MAGNITUDE IS TOO SMALL)

THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER UPON THE ASSIGNMENT OF A NONZERO CONSTANT WHICH IS TOO SMALL FOR THE IMPLEMENTATION A ZERO WILL BE SUPPLIED.

BEGIN TEST.

IF THE NUMBER PRINTED AFTER THIS STATEMENT IS ZERO, THEN THE TEST WILL HAVE PASSED.
0

62.0 EXCEPTION TEST - ON THE MAGNITUDE OF A NONZERO NUMERIC CONSTANT THAT IS TOO LARGE

62.1 Positive Machine Infinity

This test assigns a numerical constant 9.99999E99999 to a simple variable. It requires the implementation-supplied machine infinity be printed. But, preceding the output of the machine infinity, there should appear a printed message informing the user what sign to look for preceding the printed constant. The reader is referred to section 7.5 of BSR X3.60.

62.2 Negative Machine Infinity

This test uses the numerical constant (-999999)E99999 assigned to a simple variable. On output, the test requires that the negative implementation-supplied machine infinity be supplied and printed. Before this value is printed, however, there should be an informative message to the user as to what value should be printed in order that the user can judge whether the implementation fails or passes the test.

```
********************
* PROGRAM FILE 62 *
********************
```

```
0010 PRINT "PROGRAM FILE 62"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 62.1"
0100 PRINT
0110 PRINT "   (THE MAGNITUDE IS TOO LARGE, POSITIVE MACHINE INFINITY.)"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 LET A=9.99999E99999
0170 PRINT "      IF THE NUMBER PRINTED AFTER THIS STATEMENT IS POSITIVE"
0180 PRINT "AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST"
0190 PRINT "WILL HAVE PASSED."
0200 PRINT A
0210 PRINT
0220 PRINT "                                    END TEST."
0230 PRINT
0240 PRINT
0250 PRINT
0260 PRINT "                          SECTION 62.2"
```

```
0270 PRINT
0280 PRINT "   (THE MAGNITUDE IS TOO LARGE, NEGATIVE MACHINE INFINITY.)"
0290 PRINT
0300 PRINT
0310 PRINT "                                    BEGIN TEST."
0320 PRINT
0330 LET A=-9.99999E99999
0340 PRINT "      IF THE NUMBER PRINTED AFTER THIS STATEMENT IS NEGATIVE"
0350 PRINT "AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST"
0360 PRINT "WILL HAVE PASSED."
0370 PRINT A
0380 PRINT
0390 PRINT "                                      END TEST."
0400 PRINT
0410 END
```

****
* SAMPLE OUTPUT *
****

```
PROGRAM FILE 62

                         SECTION 62.1

   (THE MAGNITUDE IS TOO LARGE, POSITIVE MACHINE INFINITY.)

                                BEGIN TEST.

?OVERFLOW IN LINE 160

     IF THE NUMBER PRINTED AFTER THIS STATEMENT IS POSITIVE
AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST
WILL HAVE PASSED.
 1.70141E+38

                                END TEST.


                         SECTION 62.2

   (THE MAGNITUDE IS TOO LARGE, NEGATIVE MACHINE INFINITY.)

                                BEGIN TEST.
```

```
?OVERFLOW IN LINE 330

    IF THE NUMBER PRINTED AFTER THIS STATEMENT IS NEGATIVE
AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST
WILL HAVE PASSED.
-1.70141E+38

                                END TEST.
```

63.0 DIM STATEMENT WITH THE OPTION STATEMENT

The objective of this test is to verify that, in using the OPTION-statement with a lower bound of 1, implementations will recognize any subscript value less than 1 as an exception. In this test arrays are explicitly dimensioned. The OPTION feature is in general not available in existing processors. As a result, a user may obtain a diagnostic referencing an illegal statement in line 255. For new processors with this feature, there should only be output diagnostics as in the sample output. A diagnostic is required for a subscript of range as specified in section 6.5 of BSR X3.60. For other specifications the user is referred to section 15 of BSR X3.60.

There should be an implementation-defined diagnostic for attempting to access the zero subscript element in lines 400 and 440. However, execution of this program may terminate with a diagnostic reference to line 400 only. Should the implementation fail to recognize the error, the test has a message printed which will inform the user that the implementation has failed to recognize the subscripted error.

```
********************
* PROGRAM FILE 63 *
********************
```

```
0010 PRINT "PROGRAM FILE 63"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                                SECTION 63.0"
0100 PRINT
0110 PRINT "               (DIM-STATEMENT WITH THE OPTION-STATEMENT.)"
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "*****NOTE:  THE OBJECTIVE OF THIS PART IS TO DETERMINE"
0160 PRINT "WHETHER THE SYSTEM RECOGNIZES WHEN AN UPPER BOUND OF ZERO"
0170 PRINT "IS SPECIFIED FOR A SUBSCRIPT AS A FATAL ERROR (THAT IS,"
0180 PRINT "SUSPENDING PROGRAM EXECUTION PENDING USER-DIRECTED RESTART"
0190 PRINT "PROCEDURES) WHEN AN OPTION-STATEMENT SPECIFIES THAT ALL"
0200 PRINT "LOWER BOUNDS ARE ONE.*****"
0210 PRINT
0220 PRINT
0230 PRINT
0240 PRINT "                                   BEGIN TEST."
0250 PRINT
0255 OPTION BASE 1
0260 DIM B(15,15),C(25)
0280 LET B1=0
```

```
0290 LET B2=0
0300 LET S1=0
0310 LET S2=0
0320 FOR I=1 TO 15
0330 LET B(I,12)=B1^2
0340 LET B1=B1+2
0350 NEXT I
0360 FOR I=1 TO 15
0370 LET S1=S1+B(I,12)
0380 NEXT I
0390 FOR I=0 TO 25
0400 LET C(I)=(B2+I)^2
0410 LET B2=B2+1
0420 NEXT I
0430 FOR I=0 TO 25
0440 LET S2=S2+C(I)
0450 NEXT I
0460 PRINT "     WHETHER THERE ARE/ARE NOT ANY NUMERALS PRINTED BELOW"
0470 PRINT "THIS STATEMENT, THE SYSTEM HAS FAILED THE TEST."
0480 PRINT S1,S2
0490 PRINT
0500 PRINT "                                   END TEST."
0510 PRINT
0520 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 63

SECTION 63.0

(DIM-STATEMENT WITH THE OPTION-STATEMENT.)

*****NOTE: THE OBJECTIVE OF THIS PART IS TO DETERMINE WHETHER THE SYSTEM RECOGNIZES WHEN AN UPPER BOUND OF ZERO IS SPECIFIED FOR A SUBSCRIPT AS A FATAL ERROR (THAT IS, SUSPENDING PROGRAM EXECUTION PENDING USER-DIRECTED RESTART PROCEDURES) WHEN AN OPTION-STATEMENT SPECIFIES THAT ALL LOWER BOUNDS ARE ONE.*****

BEGIN TEST.

```
? ARRAY INDEX OUT-OF-BOUNDS IN LINE 400
```

64.0 USING THE OPTION BASE-STATEMENT TO CHANGE IMPLICIT ARRAY LOWER BOUNDS

We know from previous tests that a program, written without OPTION BASE, using implicitly dimensioned arrays, will have 0 as a lower bound for the arrays. However, when OPTION BASE is introduced, we can increase the lower bound of the arrays to 1. Thus, in order to test that this is so, we must attempt to access the 0-th element of an array in a program with the declaration OPTION BASE 1. A diagnostic is required by the standard, since a subscript of 0 would be out of bounds for the arrays. Processors may flag line 80 as an illegal statement if they do not recognize the OPTION-statement.

****
* PROGRAM FILE 64 *
****

```
0010 PRINT "PROGRAM FILE 64"
0020 PRINT
0030 PRINT
0040 PRINT
0050 PRINT "  SECTION 64.0:  OPTION BASE WITH IMPLICIT DIMENSIONING"
0060 PRINT
0070 PRINT "                                  BEGIN TEST."
0080 OPTION BASE 1
0090 FOR I=10 TO 0 STEP -1
0100 LET A(I)=I
0110 LET B(I,I)=I
0120 NEXT I
0130 PRINT
0140 PRINT " A(0)=";A(0),"B(0,0)=";B(0,0)
0150 PRINT
0160 PRINT " IF A(0)=0 AND B(0,0)=0, THEN THE OPTION BASE STATEMENT DID"
0170 PRINT " NOT AFFECT THE DEFAULT LOWER BOUND OF 0.  TEST FAILED."
0180 PRINT
0190 PRINT "                               END TEST."
0200 PRINT
0210 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 64

SECTION 64.0: OPTION BASE WITH IMPLICIT DIMENSIONING

BEGIN TEST.

? ARRAY INDEX OUT-OF-BOUNDS IN LINE 100

65.0 TESTING THE ASSIGNMENT OF ZERO FOR AN EXPRESSION CAUSING UNDERFLOW UPON EVALUATION

The objective of this test is to verify that the implementation will assign a value of zero to an expression that causes an underflow. In this case a simple variable is assigned a numerical value generated by raising 999999 to the -99999 power. This value is too small to be represented in general. On output, the test should print zero, after informing the user of this expected value. The reader is referred to section 7.4 of BSR X3.60 for this section.

****
* PROGRAM FILE 65 *
****

```
0010 PRINT "PROGRAM FILE 65"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "SECTION 65.0: TEST FOR ASSIGNMENT OF ZERO FOR AN EXPRESSION"
0100 PRINT "               WHICH CAUSES UNDERFLOW UPON EVALUATION."
0110 PRINT
0120 PRINT
0130 PRINT
0140 PRINT "                                BEGIN TEST."
0150 PRINT
0160 LET X=999999
0170 LET X=999999^-99999
0180 PRINT "     IF ZERO IS PRINTED BELOW THIS STATEMENT THEN THE SYS-"
0190 PRINT "TEM WILL HAVE PASSED THE TEST."
0200 PRINT X
0210 PRINT
0220 PRINT "                                  END TEST."
0230 PRINT
0240 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 65

SECTION 65.0: TEST FOR ASSIGNMENT OF ZERO FOR AN EXPRESSION
WHICH CAUSES UNDERFLOW UPON EVALUATION.

BEGIN TEST.

IF ZERO IS PRINTED BELOW THIS STATEMENT THEN THE SYSTEM WILL HAVE PASSED THE TEST.
0

END TEST.

66.0 GOSUB/RETURN-STATEMENT

This test unit verifies the relationship between the GOSUB-statement and the RETURN-statement. These statement types allow subroutines to be written within a program. These subroutines differ from user-defined functions because they in general might produce more complicated results than a single value as the function routine would.

The action of the GOSUB and RETURN statements can be described in terms of a stack concept. Prior to execution of the first GOSUB-statement by the test, the stack should be empty. Each time a GOSUB-statement is executed, the line number of the GOSUB-statement should be placed on top of the stack and execution of the program should continue at the line specified in the GOSUB-statement. Each time a RETURN-statement is executed, the line number on top of the stack should be removed from the stack and execution of the program should continue at the line following the one with that line number. Equal numbers of GOSUB-statements and RETURN-statements need not necessarily be executed before termination of a program. The reader should refer to section 10.4 of BSR X3.60 for the specifications.

66.1 One GOSUB and One RETURN

This test verifies that a GOSUB-statement and a RETURN-statement perform together. The control of the GOSUB-statement and the RETURN-statement is checked by a counter, N. If the control action proves to be proper, the value of N should be 2 at the termination of the test. On output, there should be only one of two possible printed messages. These are statements to the effect that the test either failed or passed. If the test fails, then the following message should be printed: RELATION BETWEEN GOSUB/RETURN, FAILED TEST. If the test passes, then the following message should be printed: RELATION BETWEEN GOSUB/RETURN, PASSED TEST.

66.2 Two GOSUB Statements Before a RETURN

The objective of this test is to verify that two GOSUB-statements can be executed without an intervening RETURN-statement. Through the use of a counter, N, the performance of the two GOSUB-statements is checked as well as the performance of the RETURN-statement in conjunction with the last GOSUB-statement. Upon proper performance of the two GOSUB-statements and the RETURN-statement, the value of the counter should be 3. The output for this test should be a message indicating pass or fail. If the test fails, then the following message should be printed: TWO GOSUBS WITHOUT INTERVENING RETURN, FAILED TEST. If the test passes, then the following message should be printed: TWO GOSUBS WITHOUT INTERVENING RETURN, PASSED TEST.

66.3 Testing Proper GOSUB Returns

The purpose of this test is to verify the stack-like relationship between GOSUB-statements and RETURN-statements through the use of nested GOSUB-statements with RETURN-statements. There are four levels of nesting performed by this test. The number of GOSUB-statements per level is equal to the number of its level. For each level of GOSUB-statements, there is only one RETURN-statement. This program also tests that an equal number of

GOSUB-statements and RETURN-statements need not necessarily be executed before termination of a program. The output for this test should either be a fail or a pass message. If the test fails, then the following message should be printed: GOSUB NESTING, FAILED TEST. If the test passes, then the following message should be printed: GOSUB NESTING, PASSED TEST.

```
**********
* PROGRAM FILE 66 *
**********
```

```
0010 PRINT "PROGRAM FILE 66"
0020 PRINT
0030 PRINT
0040 PRINT
0070 PRINT "                SECTION 66.1: ONE GOSUB AND ONE RETURN."
0080 PRINT
0090 PRINT "                                BEGIN TEST."
0100 PRINT
0110 LET N=0
0120 GOSUB 190
0130 LET N=N+1
0140 IF N=2 THEN 170
0150 PRINT "              RELATION BETWEEN GOSUB/RETURN, FAILED TEST."
0160 GOTO 230
0170 PRINT "              RELATION BETWEEN GOSUB/RETURN, PASSED TEST."
0180 GOTO 210
0190 LET N=N+1
0200 RETURN
0210 PRINT
0220 PRINT "                                END TEST."
0230 PRINT
0240 PRINT "               SECTION 66.2: TWO GOSUBS BEFORE A RETURN."
0250 PRINT
0260 PRINT "                                BEGIN TEST."
0270 PRINT
0280 LET N=0
0530 GOSUB 550
0540 PRINT "                        ERROR, FIRST GOSUB FAILED."
0550 LET N=N+1
0560 GOSUB 630
0570 LET N=N+1
0580 IF N=3 THEN 610
0590 PRINT "    TWO GOSUBS WITHOUT AN INTERVENING RETURN, FAILED TEST."
0600 GOTO 670
0610 PRINT "    TWO GOSUBS WITHOUT AN INTERVENING RETURN, PASSED TEST."
0620 GOTO 650
0630 LET N=N+1
0640 RETURN
0650 PRINT
```

```
0660 PRINT "                                   END TEST."
0670 PRINT
0680 PRINT "              SECTION 66.3: TESTING PROPER GOSUB RETURNS"
0690 PRINT
0700 PRINT "                                   BEGIN TEST."
0710 PRINT
0720 LET N=0
0730 GOSUB 780
0740 IF N=24 THEN 940
0750 PRINT "                         GOSUB NESTING, FAILED TEST."
0760 PRINT
0770 GOTO 950
0780 GOSUB 860
0790 GOSUB 860
0800 RETURN
0810 GOSUB 900
0820 GOSUB 900
0830 GOSUB 900
0840 GOSUB 900
0850 RETURN
0860 GOSUB 810
0870 GOSUB 810
0880 GOSUB 810
0890 RETURN
0900 LET N=N+1
0910 IF N=7 THEN 890
0920 RETURN
0930 GOTO 750
0940 PRINT "                         GOSUB NESTING, PASSED TEST."
0950 PRINT
0960 PRINT "                                   END TEST."
0970 PRINT
0980 PRINT
0990 END
```

```
*****************
* SAMPLE OUTPUT *
*****************
```

```
PROGRAM FILE 66

              SECTION 66.1: ONE GOSUB AND ONE RETURN.

                            BEGIN TEST.

            RELATION BETWEEN GOSUB/RETURN, PASSED TEST.
```

```
END TEST.

SECTION 66.2: TWO GOSUBS BEFORE A RETURN.

BEGIN TEST.
TWO GOSUBS WITHOUT AN INTERVENING RETURN, PASSED TEST.
END TEST.

SECTION 66.3: TESTING PROPER GOSUB RETURNS

BEGIN TEST.
GOSUB NESTING, PASSED TEST.
END TEST.
```

67.0 SEMANTIC ERROR - TEST ON GOSUB TRANSFER TO AN ILLEGAL LINE NUMBER

The objective of this test is to verify that the implementation will recognize a transfer by a GOSUB-statement to a non-existent line as an error. The test has a GOSUB-statement which uses a non-existent program line number as its designated transfer point in line 260. Although this error is not considered an exception it is not a meaningful construction and should be handled by an implementation with a diagnostic pointing to an illegal line number in line 260. After the diagnostic the program should be terminated. On output, there should be some form of implementation-defined diagnostic. However, the test does have a message printed should the implementation fail to recognize the error or ignore the line with the error. The reader is referred to section 10.4 of BSR X3.60 for the specifications.

```
********************
* PROGRAM FILE 67 *
********************
```

```
0010 PRINT "PROGRAM FILE 67"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "      SECTION 67.0: GOSUB TO ILLEGAL LINE NUMBER"
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO USE A GOSUB-STATE-"
0140 PRINT "MENT WHICH REFERS TO A NON-EXISTENT LINE NUMBER IN ORDER TO"
0150 PRINT "DETERMINE WHETHER THE SYSTEM WILL RECOGNIZE THIS PROCEDURE"
0160 PRINT "AS A FATAL ERROR.  THAT IS, SUCH A RECOGNITION BY THE SYS-"
0170 PRINT "TEM WILL SUSPEND EXECUTION OF THE PROGRAM PENDING USER-"
0180 PRINT "DIRECTED RESTART PROCEDURES.  IF SUCH A RESULT SHOULD OCCUR,"
0190 PRINT "THEN THE TEST WILL HAVE PASSED."
0200 PRINT
0210 PRINT
0220 PRINT
0230 PRINT "                                BEGIN TEST."
0240 PRINT
0250 LET F=0
0260 GOSUB 123
0270 IF F=1 THEN 300
0280 PRINT "TEST FAILED BECAUSE GOSUB-STATEMENT WAS IGNORED."
0290 GOTO 340
0300 PRINT "TEST FAILED BECAUSE TRANSFER WAS MADE TO NON-EXISTENT LINE."
0310 GOTO 340
0320 LET F=1
0330 RETURN
```

```
0340 PRINT
0350 PRINT "                                   END TEST."
0360 PRINT
0370 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

In order for this test to pass, an error must be diagnosed and reported. A possible error diagnostic for this program might be:

```
? UNDEFINED LINE NUMBER 123 IN LINE 260
```

68.0 EXCEPTION TEST - RETURN-STATEMENT WITHOUT GOSUB

The objective of this test is to verify that attempting to execute a RETURN-statement without having executed a corresponding GOSUB-statement will be diagnosed as an exception. This requires a diagnostic message and termination of the program since there are no specified recovery procedures in the ANSI Minimal BASIC standard. On output, there should be some form of implementation-defined diagnostic describing the nature of the error. However, the test is constructed to allow, in the event that the implementation fails to recognize the error, the output of a message that will inform the user that the implementation failed the test. The reader is referred to section 10.5 of BSR X3.60.

```
********************
* PROGRAM FILE 68 *
********************
```

```
0010 PRINT "PROGRAM FILE 68"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "      SECTION 68.0: FATAL ERROR CHECK ON RETURN-STATEMENT."
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO EXECUTE A RETURN-"
0140 PRINT "STATEMENT WITHOUT HAVING EXECUTED A CORRESPONDING GOSUB-"
0150 PRINT "STATEMENT SO THAT IT MAYBE DETERMINED WHETHER SUCH AN EXE-"
0160 PRINT "CUTION IS PERMISSIBLE BY THIS SYSTEM.  IF THE SYSTEM SHOULD"
0170 PRINT "RECOGNIZE THIS EXECUTION AS A FATAL ERROR (THAT IS, SUS-"
0180 PRINT "PENDING PROGRAM EXECUTION PENDING USER-DIRECTED RESTART"
0190 PRINT "PROCEDURES), THEN THE TEST WILL HAVE PASSED."
0200 PRINT
0210 PRINT
0220 PRINT
0230 PRINT "                                BEGIN TEST."
0240 PRINT
0250 RETURN
0260 PRINT "SYSTEM FAILED TEST."
0270 PRINT
0280 PRINT "                                 END TEST."
0290 PRINT
0300 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 68

SECTION 68.0: FATAL ERROR CHECK ON RETURN-STATEMENT.

THE OBJECTIVE OF THIS SECTION IS TO EXECUTE A RETURN-STATEMENT WITHOUT HAVING EXECUTED A CORRESPONDING GOSUB-STATEMENT SO THAT IT MAYBE DETERMINED WHETHER SUCH AN EXECUTION IS PERMISSIBLE BY THIS SYSTEM. IF THE SYSTEM SHOULD RECOGNIZE THIS EXECUTION AS A FATAL ERROR (THAT IS, SUSPENDING PROGRAM EXECUTION PENDING USER-DIRECTED RESTART PROCEDURES), THEN THE TEST WILL HAVE PASSED.

BEGIN TEST.

? RETURN BEFORE GOSUB IN LINE 250

69.0 TESTING ROUNDOFF TO SIX SIGNIFICANT DIGITS OF CONSTANTS OF ARBITRARY LENGTH

The objective of this test is to verify that although the accuracy, with which evaluation of an expression takes place, varies from implementation to implementation, each implementation should attempt to maintain at least six decimal digits of precision. For each test the output should contain a minimum of six significant digits. Furthermore, programs can contain numeric constants of an arbitrary number of digits, although an implementation may choose to round them to no less than six significant digits. The reader is referred to section 5.4 of BSR X3.60.

69.1 Using Numerically Assigned Constants of Six or Fewer Significant Digits

The objective of this test is to verify, for various numerical operations, that the implementation will maintain at least six decimal digits of precision. This test uses constants of six digits or fewer. Although rounding has been tested before for operations on numbers made up of less than or equal to six significant digits, this part of the test is included for completeness.

The test has a three column formatted output. In the first column, titled "True Rounded Values", there should be a list of the expected rounded values. In the second column, titled "System Rounded Values", there should be a list of the system evaluations, as rounded by the implementation. In the third column, titled "Absolute Error", there should be the listings of marginal differences between the expected rounded values and the respective implementation rounded values. If any value in the third column does not fall within the expected or allowed range of one unit error in the position of the sixth significant digit, then an asterisk should have appeared beside that difference.

69.2 Using Numerically Assigned Constants of More Than Six Digits of Significance

The objective of this test is to verify that the implementation will maintain at least six decimal digits of precision for numbers with an arbitrary number of digits of precision. This test uses assignment of constants that are composed of up to 17 decimal digits of precision. These assignments are then used in various operations. On output, this test has the same output format described in section 69.1.

****
* PROGRAM FILE 69 *
****

```
0010 PRINT "PROGRAM FILE 69"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 69.0: ROUNDOFF."
0100 PRINT
0110 PRINT "              (TESTING ROUNDOFF TO SIX SIGNIFICANT DIGITS.)"
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "     IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR"
0160 PRINT "COLUMN, TEST PASSED.  HOWEVER, IF AN ASTERISK FOLLOWS"
0170 PRINT "A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BE-"
0180 PRINT "CAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF."
0190 PRINT
0200 PRINT
0210 PRINT " TRUE  ","SYSTEM "
0220 PRINT "ROUNDED","ROUNDED","ABSOLUTE"
0230 PRINT " VALUE "," VALUE "," ERROR  "
0240 PRINT "************************************************";
0250 PRINT "**********************"
0260 PRINT
0270 PRINT
0280 PRINT "                          SECTION 69.1"
0290 PRINT
0300 PRINT "*****USING NUMERICALLY ASSIGNED CONSTANTS OF SIX OR LESS SIG-"
0310 PRINT "NIFICANT DIGITS.*****"
0320 PRINT
0330 PRINT
0340 PRINT "                                  BEGIN TEST."
0350 PRINT
0360 DIM A(12)
0370 LET A(1)=3.74959
0380 LET A(2)=1.E28
0390 LET A(3)=1.E-16
0400 LET A(4)=9.99999E-37
0410 LET A(5)=9.99888E-17
0420 LET F=0
0430 LET A$=" "
0440 LET B$=" 9.99888 "
0450 LET A(6)=A(1)*2.66666
0460 LET B=9.99888
0470 LET E=A(6)-B
0480 IF ABS(E)<=1E-5 THEN 500
0490 LET A$="*"
0500 GOSUB 1000
0510 LET A$=" "
0520 LET B$=" 3.74959 "
0530 LET A(7)=B/2.66666
0540 LET C=A(1)
0550 LET E=A(7)-C
0560 IF ABS(E)<=1E-5 THEN 580
0570 LET A$="*"
0580 GOSUB 1000
0590 LET A$=" "
0600 LET B$=" 9.00000E37 "
0610 LET A(8)=A(2)*9.E9
```

```
0620 LET B=9.E37
0630 LET E=A(8)-B
0640 IF ABS(E)<=1E32 THEN 660
0650 LET A$="*"
0660 GOSUB 1000
0670 LET A$=" "
0680 LET B$=" 2.62144E-33 "
0690 LET A(9)=A(3)*.262144E-16
0700 LET B=2.62144E-33
0710 LET E=A(9)-B
0720 IF ABS(E)<=1E-38 THEN 740
0730 LET A$="*"
0740 GOSUB 1000
0750 LET A$=" "
0760 LET B$=" 99.9998 "
0770 LET A(10)=A(4)*9.99999E37
0780 LET B=99.9998
0790 LET E=A(10)-B
0800 IF ABS(E)<=1E-4 THEN 820
0810 LET A$="*"
0820 GOSUB 1000
0830 LET A$=" "
0840 LET B$=" 2.66666E-33 "
0850 LET A(11)=A(5)/3.74959E16
0860 LET B=2.66666E-33
0870 LET E=A(11)-B
0880 IF ABS(E)<=1E-38 THEN 900
0890 LET A$="*"
0900 GOSUB 1000
0910 LET A$=" "
0920 LET B$=" 524288 "
0930 LET A(12)=524287+1
0940 LET B=524288
0950 LET E=A(12)-B
0960 IF ABS(E)<=1E0 THEN 980
0970 LET A$="*"
0980 GOSUB 1000
0990 GOTO 1060
1000 IF F<>0 THEN 1020
1010 LET I=6
1020 PRINT B$,A(I),E;A$
1030 LET I=I+1
1040 LET F=1
1050 RETURN
1060 PRINT
1070 PRINT "                                  END TEST."
1080 PRINT
1090 PRINT
1100 PRINT
1110 PRINT "                            SECTION 69.2"
1120 PRINT
1130 PRINT "*****USING NUMERICALLY ASSIGNED CONSTANTS OF MORE THAN SIX"
1140 PRINT "DIGITS OF SIGNIFICANCE.*****"
1150 PRINT
1160 PRINT
1170 PRINT "                                BEGIN TEST."
1180 PRINT
```

```
1190 LET A(1)=3.749586439134E0
1200 LET A(2)=.15707963267948966E+28
1210 LET A(3)=.100047836917365S7E-34
1220 LET A(4)=9.9999999999999996E-36
1230 LET A(5)=9.9996666866652382E-17
1240 LET F=0
1250 LET A$=" "
1260 LET B$=" 9.99889 "
1270 LET A(6)=A(1)*2.6666635278931E0
1280 LET B=9.99889
1290 LET E=A(6)-B
1300 IF ABS(E)<=1E-5 THEN 1320
1310 LET A$="*"
1320 GOSUB 1820
1330 LET A$=" "
1340 LET B$=" 3.74959 "
1350 LET A(7)=9.998885402/2.6666635278931E0
1360 LET C=3.74959
1370 LET E=A(7)-C
1380 IF ABS(E)<=1E-5 THEN 1400
1390 LET A$="*"
1400 GOSUB 1820
1410 LET A$=" "
1420 LET B$=" 1.05988E32 "
1430 LET A(8)=A(2)*.67474094222355266E5
1440 LET B=1.05988E32
1450 LET E=A(8)-B
1460 IF ABS(E)<=1E27 THEN 1480
1470 LET A$="*"
1480 GOSUB 1820
1490 LET A$=" "
1500 LET B$=" 3.73503E-33 "
1510 LET A(9)=.37332419967990016E3 * A(3)
1520 LET B=3.73503E-33
1530 LET E=A(9)-B
1540 IF ABS(E)<=1E-38 THEN 1560
1550 LET A$="*"
1560 GOSUB 1820
1570 LET A$=" "
1580 LET B$=" 100 "
1590 LET A(10)=A(4)*9.9999999999999996E36
1600 LET B=100
1610 LET E=A(10)-B
1620 IF ABS(E)<=1E-3 THEN 1640
1630 LET A$="*"
1640 GOSUB 1820
1650 LET A$=" "
1660 LET B$=" 3.81958E-33 "
1670 LET A(11)=A(5)/2.6180014127101748E16
1680 LET B=3.81958E-33
1690 LET E=A(11)-B
1700 IF ABS(E)<=1E-38 THEN 1720
1710 LET A$="*"
1720 GOSUB 1820
1730 LET A$=" "
1740 LET B$=" 3.0376 "
1750 LET A(12)=.14801364395941515E1+.15574637835007509E1
```

```
1760 LET B=3.0376
1770 LET E=A(12)-B
1780 IF ABS(E)<=1E-5 THEN 1800
1790 LET A$="*"
1800 GOSUB 1820
1810 GOTO 1880
1820 IF F<>0 THEN 1840
1830 LET I=6
1840 PRINT B$,A(I),E;A$
1850 LET I=I+1
1860 LET F=1
1870 RETURN
1880 PRINT
1890 PRINT "                                   END TEST."
1900 PRINT
1910 PRINT
1920 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 69

SECTION 69.0: ROUNDOFF.

(TESTING ROUNDOFF TO SIX SIGNIFICANT DIGITS.)

IF NO ASTERISK FOLLOWS ANY VALUE IN THE ABSOLUTE ERROR COLUMN, TEST PASSED. HOWEVER, IF AN ASTERISK FOLLOWS A VALUE IN THE ABSOLUTE ERROR COLUMN, TEST FAILED BECAUSE SYSTEM WOULD HAVE FAILED THE ERROR BOUND ROUND-OFF.

TRUE ROUNDED VALUE	SYSTEM ROUNDED VALUE	ABSOLUTE ERROR

SECTION 69.1

*****USING NUMERICALLY ASSIGNED CONSTANTS OF SIX OR LESS SIGNIFICANT DIGITS.*****

BEGIN TEST.

9.99888	9.99888	0
3.74959	3.74959	0
9.00000E37	9.00000E+37	0
2.62144E-33	2.62144E-33	0
99.9998	99.9998	0
2.66666E-33	2.66666E-33	0
524288	524288	0

END TEST.

SECTION 69.2

*****USING NUMERICALLY ASSIGNED CONSTANTS OF MORE THAN SIX DIGITS OF SIGNIFICANCE.*****

BEGIN TEST.

9.99889	9.99889	0
3.74959	3.74959	0
1.05988E32	1.05988E+32	0
3.73503E-33	3.73503E-33	0
100	100	0
3.81958E-33	3.81958E-33	0
3.0376	3.0376	0

END TEST.

70.0 THE ON-GOTO STATEMENT

This test verifies that the ON-GOTO statement for the test system can round its numeric expression to an integer and use that integer to select the appropriate line number from a list of line numbers following the GOTO. In particular, suppose that there are N line numbers. Then, if the statement expression is rounded to an integer M, this integer is either less than 1, one of the integers from 1 to N, or greater than N. If M is one of the integers 1 to N then the control statement transfers control to the M-th line number in the list. Otherwise, the system must report an exception. The reader is referred to section 10 of BSR X3.60.

70.1 The ON-GOTO Numeric Expression, Using an Integer Within Range

The objective of this test is to show that the conditional transfer should be performed properly if the simple variable I, used in the ON-GOTO expression, is an integer from 1 to 5. 5 is the list length. I is also the simple variable of a FOR-NEXT loop which uses integers for its initial value and limit, and has no STEP clause. The test informs the user when the transfer was not made by the ON-GOTO-statement to the correct statement. If there is no transfer, then the following message should be printed: THE <number> ON-GOTO TRANSFER, FAILED. Then, the following message should be printed: ERROR, TRANSFER SHOULD HAVE BEEN TO LINE NO. <number> IN LIST. That the transfer was actually made to the correct statement is determined by an IF-THEN statement. This tests the FOR-NEXT loop index. There is a counter that acts as a bookkeeper for the number of correct transfers, which in this case should be five. If this counter is not five, then the following message should be printed: ON-GOTO-STATEMENT, FAILED TEST. If the counter is five, then the following message should be printed: ON-GOTO-STATEMENT, PASSED TEST.

70.2 The ON-GOTO Numeric Expression, As a Fraction Rounded to an Integer

This test determines the round-off capability of the ON-GOTO-statement for numeric expressions. The Minimal BASIC standard requires rounding of the expression value to the nearest integer before performing the transfer. The numeric expression in this test is the simple control variable of a FOR-NEXT loop in which the initial value is incremented in steps of .5 to the limit. K counts the number of passes through the loop. In this case there should be 5. On the first pass (K=1), transfer should be to the first ON-GOTO line number. On the second and third, the transfer should be to the second line number and finally, on the fourth and fifth passes, the transfer should be to the third line number. There is then one transfer to the first number and two each to the second and third line numbers. These counts are tested by the variables A, B, and C, respectively. If the ON-GOTO statement fails and the program continues, then a message follows: THE <number> ON-GOTO TRANSFER, FAILED. The second possible message is as follows: ERROR, TRANSFER SHOULD HAVE BEEN TO LINE NO. <number> IN LIST. Finally, the variables A, B, and C are each used to keep count of the number of transfers made by the ON-GOTO statement to the line numbers in the ON-GOTO list. The values of the correct number of transfers by A, B, and C should be 1, 2, and 2 respectively. Proper transfers by the ON-GOTO-statement are determined by IF-THEN-statements which are placed at each of the line numbers of the

ON-GOTO list. If upon completion of the FOR-NEXT loopings, the values of the counters A, B, and C are not 1, 2, and 2 respectively, the following message should be printed: ON-GOTO-STATEMENT, FAILED TEST. If the value of the counters A, B, and C are in order, then the following message should be printed: ON-GOTO-STATEMENT, PASSED TEST.

70.3 The ON-GOTO Numeric Expression, As An Expression of More Than One Term

This test verifies the proper evaluation and use for transfer control of an expression of more than one term by the ON-GOTO-statement. For this test, as for the previous tests, the evaluation should be based on the nearest integer value of the expression. Through the use of FOR-NEXT loops, different values are assigned for the evaluation of the numeric expression by the ON-GOTO-statement. The values of the numeric expression should be 1, 2, 3, 4, and 5. These values should be the position indices of the line numbers in the ON-GOTO list. Within the FOR-NEXT loop, there are two checks on the ON-GOTO-statement, and two counters. The first counter again keeps count of which transfer is being made and is used in the first error message. In the second error message, it acts as a pointer to the correct line number in the ON-GOTO list. The two messages should be the same as in the past two tests. Upon completion of the FOR-NEXT loops, as a means for checking the proper performance of the ON-GOTO-statement, the value of the second counter is checked for a value of 5. If the value of the count is not five, then the following message should be printed: ON-GOTO-STATEMENT, FAILED TEST. If the value of the count is five, then the following message should be printed: ON-GOTO-STATEMENT, PASSED TEST.

```
**********
* PROGRAM FILE 70 *
**********
```

```
0010 PRINT "PROGRAM FILE 70"
0020 PRINT
0030 PRINT
0040 PRINT
0050 PRINT "            SECTION 70.0: THE ON-GOTO STATEMENT"
0080 PRINT
0090 PRINT "        SECTION 70.1: THE ON-GOTO NUMERIC EXPRESSION"
0100 PRINT
0110 PRINT "              USING AN INTEGER WITHIN RANGE"
0120 PRINT
0130 PRINT "                                  BEGIN TEST."
0140 PRINT
0150 LET K=0
0160 LET N=0
0170 FOR I=1 TO 5
0180 LET K=K+1
```

```
0190 ON I GOTO 220,240,260,280,300
0200 PRINT TAB(20);"THE";K;"ON-GOTO TRANSFER, FAILED."
0210 PRINT
0220 IF I=1 THEN 320
0230 GOTO 340
0240 IF I=2 THEN 320
0250 GOTO 340
0260 IF I=3 THEN 320
0270 GOTO 340
0280 IF I=4 THEN 320
0290 GOTO 340
0300 IF I=5 THEN 320
0310 GOTO 340
0320 LET N=N+1
0330 GOTO 350
0340 PRINT "ERROR, TRANSFER SHOULD HAVE BEEN TO LINE NO.";K;"IN LIST."
0350 NEXT I
0360 PRINT
0370 IF N=5 THEN 400
0380 PRINT "                    ON-GOTO-STATEMENT, FAILED TEST."
0390 GOTO 430
0400 PRINT "                    ON-GOTO-STATEMENT, PASSED TEST."
0410 PRINT
0420 PRINT "                                END TEST."
0430 PRINT
0440 PRINT "          SECTION 70.2: THE ON-GOTO NUMERIC EXPRESSION"
0450 PRINT
0455 PRINT "              AS A FRACTION ROUNDED TO AN INTEGER"
0457 PRINT
0460 PRINT "                                BEGIN TEST."
0470 PRINT
0480 LET A=0
0490 LET B=0
0500 LET C=0
0510 LET K=0
0520 FOI I=1 TO 3 STEP .5
0530 LET K=K+1
0540 ON I GOTO 570,590,620
0550 PRINT TAB(20);"THE";K;"ON-GOTO TRANSFER, FAILED."
0560 PRINT
0570 IF I=1 THEN 730
0580 GOTO 640
0590 IF I=1.5 THEN 750
0600 IF I=2.0 THEN 750
0610 GOTO 640
0620 IF I=2.5 THEN 770
0630 IF I=3.0 THEN 770
0640 IF K=1 THEN 680
0650 IF K>=4 THEN 700
0660 LET F=2
0670 GOTO 710
0680 LET F=1
0690 GOTO 710
0700 LET F=3
0710 PRINT "ERROR, TRANSFER SHOULD HAVE BEEN TO LINE NO.";F;"IN LIST."
0720 GOTO 780
0730 LET A=A+1
```

```
0740 GOTO 780
0750 LET B=B+1
0760 GOTO 780
0770 LET C=C+1
0780 NEXT I
0790 PRINT
0800 IF A=1 THEN 820
0810 GOTO 850
0820 IF B=2 THEN 840
0830 GOTO 850
0840 IF C=2 THEN 870
0850 PRINT "                    ON-GOTO-STATEMENT, FAILED TEST."
0860 GOTO 900
0870 PRINT "                    ON-GOTO-STATEMENT, PASSED TEST."
0880 PRINT
0890 PRINT "                                 END TEST."
0900 PRINT
0910 PRINT "        SECTION 70.3: THE ON-GOTO NUMERIC EXPRESSION"
0913 PRINT
0915 PRINT "           AS AN EXPRESSION OF MORE THAN ONE TERM"
0920 PRINT
0930 PRINT "                                 BEGIN TEST."
0940 PRINT
0950 LET K=0
0960 LET N=0
0970 FOR I=1 TO 5
0980 ON (3*I-2)-2*(I-1) GOTO 1010,1030,1050,1070,1090
0990 PRINT TAB(20);"THE";K;"ON-GOTO TRANSFER, FAILED."
1000 PRINT
1010 IF I=1 THEN 1110
1020 GOTO 1130
1030 IF I=2 THEN 1110
1040 GOTO 1130
1050 IF I=3 THEN 1110
1060 GOTO 1130
1070 IF I=4 THEN 1110
1080 GOTO 1130
1090 IF I=5 THEN 1110
1100 GOTO 1130
1110 LET N=N+1
1120 GOTO 1140
1130 PRINT "ERROR, TRANSFER SHOULD HAVE BEEN TO LINE NO.";K;"IN LIST."
1140 NEXT I
1150 PRINT
1160 IF N=5 THEN 1190
1170 PRINT "                    ON-GOTO-STATEMENT, FAILED TEST."
1180 GOTO 1220
1190 PRINT "                    ON-GOTO-STATEMENT, PASSED TEST."
1200 PRINT
1210 PRINT "                                 END TEST."
1220 PRINT
1230 PRINT
1240 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

```
PROGRAM FILE 70

     SECTION 70.0: THE ON-GOTO STATEMENT
   SECTION 70.1: THE ON-GOTO NUMERIC EXPRESSION
         USING AN INTEGER WITHIN RANGE
                    BEGIN TEST.
              ON-GOTO-STATEMENT, PASSED TEST.
                    END TEST.
     SECTION 70.2: THE ON-GOTO NUMERIC EXPRESSION
         AS A FRACTION ROUNDED TO AN INTEGER
                    BEGIN TEST.
              ON-GOTO-STATEMENT, PASSED TEST.
                    END TEST
   SECTION 70.3: THE ON-GOTO NUMERIC EXPRESSION
       AS AN EXPRESSION OF MORE THAN ONE TERM
                    BEGIN TEST.
              ON-GOTO-STATEMENT, PASSED TEST.
                    END TEST.
```

71.0 SEMANTIC DIAGNOSTIC - ON-GOTO STATEMENT REFERRING TO A NON-EXISTENT LINE NUMBER

This test verifies that the implementation will recognize when a transfer to an illegal line number is attempted by the ON-GOTO statement. In particular, the objective is to determine whether an attempted transfer of this sort in line 380 will be considered by the implementation as an error requiring a diagnostic and program termination. On output, there should be some form of implementation-defined diagnostic. However, should the implementation fail to recognize this error, the program prints a message which tells the user that the implementation failed the test. The reader is referred to section 10.4 of BSR X3.60.

```
********************
* PROGRAM FILE 71 *
********************
```

```
0010 PRINT "PROGRAM FILE 71"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "     SECTION 71.0: FATAL ERROR TEST - ON-GOTO-STATEMENT."
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0140 PRINT "THIS SYSTEM RECOGNIZES THE FOLLOWING AS FATAL ERRORS (THAT"
0150 PRINT "IS, SUSPENDING PROGRAM EXECUTION PENDING USER-DIRECTED RE-"
0160 PRINT "START PROCEDURES):"
0170 PRINT
0180 PRINT "     (1) WHEN AN ON-GOTO-STATEMENT REFERS TO A NON-EXISTENT"
0190 PRINT "         LINE NUMBER, OR"
0200 PRINT "     (2) WHEN THE INTEGER OBTAINED AS THE VALUE OF AN EX-"
0210 PRINT "         PRESSION IN AN ON-GOTO-STATEMENT IS LESS THAN ONE"
0220 PRINT "         OR GREATER THAN THE NUMBER OF LINE NUMBERS IN THE"
0230 PRINT "         LIST."
0240 PRINT
0250 PRINT "IF BOTH OF THE ABOVE REFERRALS ARE CONSIDERED FATAL ERRORS,"
0260 PRINT "THEN THE TEST WILL HAVE PASSED."
0270 PRINT
0280 PRINT
0290 PRINT
0310 PRINT
0320 PRINT "              (A REFERRAL TO A NON-EXISTENT LINE NUMBER.)"
0330 PRINT
0340 PRINT
0350 PRINT "                                BEGIN TEST."
0360 PRINT
```

```
0370 LET X=1
0380 ON X GOTO 159
0390 PRINT "SYSTEM FAILED TEST."
0400 PRINT
0410 PRINT "                                    END TEST."
0420 PRINT
0430 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

In order for this test to pass, a fatal error must be diagnosed and reported. A possible error diagnostic for this program might be:

```
? UNDEFINED LINE NUMBER 159 IN LINE 380
```

72.0 EXCEPTION TEST - VALUE OF ON-GOTO EXPRESSION LESS THAN ONE

This test verifies that the implementation recognizes the numeric expression evaluation with values less than one as an exception. The test has an ON-GOTO-statement which uses an expression that should round to an integer less than one at line 170. In this case the expression is a simple variable with a value of .3 that should be rounded to 0. On output, there should be some form of implementation-defined diagnostic. However, the test is structured to print a message of the implementation's failure should the test system not recognize the error. The reader is referred to section 10.5 of BSR X3.60.

****
* PROGRAM FILE 72 *
****

```
0010 PRINT "PROGRAM FILE 72"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                                SECTION 72.0"
0100 PRINT
0110 PRINT "               (VALUE OF ON-GOTO EXPRESSION LESS THAN ONE.)"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 LET X=.3
0170 ON X GOTO 190
0180 PRINT
0190 PRINT "SYSTEM FAILED TEST."
0200 PRINT
0210 PRINT "                                   END TEST."
0220 PRINT
0230 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 72

SECTION 72.0

(VALUE OF ON-GOTO EXPRESSION LESS THAN ONE.)

BEGIN TEST.

? ON EVALUATED OUT OF RANGE IN LINE 170

73.0 EXCEPTION TEST - VALUE OF ON-GOTO EXPRESSION GREATER THAN THE NUMBER OF LINE NUMBERS IN THE LIST

The objective of this test is to compute an integer value for the numerical expression used in an ON-GOTO-statement. In this case the integer value should be greater than the number of line numbers listed in the ON-GOTO-statement. The test determines whether the implementation recognizes this as an exception. The program has a simple variable, X, assigned the value 2 in line 170 but an ON-GOTO with one line number in its list in line 180. Then, X is used as the expression in line 180. On output, there should be some form of implementation-defined diagnostic relating to the error. However, should the implementation fail to recognize the error, a message will be printed to the user. The reader is referred to section 10.5 of BSR X3.60.

****
* PROGRAM FILE 73 *
****

```
0010 PRINT "PROGRAM FILE 73"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 73.0"
0100 PRINT
0110 PRINT "   (VALUE OF ON-GOTO EXPRESSION GREATER THAN THE NUMBER OF"
0115 PRINT
0120 PRINT "                     LINE NUMBERS IN THE LIST.)"
0130 PRINT
0140 PRINT
0150 PRINT "                                  BEGIN TEST."
0160 PRINT
0170 LET X=2
0180 ON X GOTO 190
0190 PRINT
0200 PRINT "SYSTEM FAILED TEST."
0210 PRINT
0220 PRINT "                                    END TEST."
0230 PRINT
0240 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 73

SECTION 73.0

(VALUE OF ON-GOTO EXPRESSION GREATER THAN THE NUMBER OF LINE NUMBERS IN THE LIST.)

BEGIN TEST.

? ON EVALUATED OUT OF RANGE IN LINE 180

74.0 READ/DATA STATEMENTS

These next eight sections are oriented towards testing (1) whether the READ-statement assigns values, provided by DATA-statements, to variables, and (2) whether the RESTORE-statement enables the rereading of those same values. The values supplied by DATA-statements can be either numeric constants, string constants, or unquoted strings. All of the data from the totality of DATA-statements should be collected together into a data sequence. It should not matter where DATA-statements are located in a program as long as they occur before the END-statement. However, the order in which the different types of data occur should determine the order of the variables within the variable list of the READ-statements. That is, the order of the numeric variables must match that of the numeric constants within the data sequence and the same for string variables. If there are variables in READ-statements with subscripted expresions, then the expressions are evaluated after values have been assigned to any variables preceding those subscripted variables (to the left of them in the list). By the use of the RESTORE-statement, the pointer associated with the data sequence should be reset to the beginning of the data sequence so that the next READ-statement executed will read data from the beginning of the sequence once again. The reader is referred to section 14 of BSR X3.60 for the specifications.

74.1 READ/DATA for Numeric Variables

The objective of this section is to introduce the READ/DATA relationship by assigning numerical constants to both simple and subscripted variables.

74.1.1 For Simple Variables

This test determines whether a list of numeric constants can be assigned to a list of simple numeric variables through the READ-statement and DATA-statement. The test has three numeric constants in the DATA-statement assigned to three simple variables by a READ. The numeric constants are in the forms NR1, NR2, and NR3. On output there should be a message flagging false assignments by the READ-statement. Each of the error messages should read as follows: READ ASSIGNMENT FOR VARIABLE NUMBER <number>, FAILED. If each variable assignment is proper then the following message should be printed: READ/DATA ASSIGNMENTS FOR SIMPLE VARIABLES, PASSED TEST.

74.1.2 For Subscripted Variables

The objective of this section is to execute READ/DATA assignments for both singly and doubly subscripted arrays.

74.1.2.1 As One-Dimensional Arrays

This test uses one-dimensional arrays for four subscripted variables. Two are assigned values by use of LET-statements. These values have also been entered into a DATA-list. The other two variables used in the READ-statement should be assigned the same values as the first two subscripted variables after the READ-statement has been executed. The values of the assignments are checked by use of IF-THEN-statements. The test also

verifies incidentally that DATA-statements can be placed anywhere in the program before the END-statement. On output, there should be an error message for any faulty READ assignments. Each of the error messages should be printed as follows: READ ASSIGNMENT FOR VARIABLE NUMBER <number>, FAILED. If each variable assignment is correct, then the following message should be printed: READ/DATA ASSIGNMENTS FOR ONE-DIMENSIONAL ARRAYS, PASSED.

74.1.2.2 As Two-Dimensional Arrays

In this test, two-dimensional arrays are used in a manner similar to that in 74.1.2.1. Two of the four subscripted variables are assigned values through the use of LET-statements. The remaining two variables, used in the READ-list, should be assigned the same values. On output there should be a printed message for any faulty READ assignment. Each of these error messages should appear as follows: READ ASSIGNMENT FOR VARIABLE NUMBER <number>, FAILED. If each assignment is correctly made, then the following message should be printed: READ/DATA ASSIGNMENTS FOR TWO-DIMENSIONAL ARRAYS, PASSED.

74.2 READ/DATA for String Variables

The object of this test is to assign string constants to string variables by using the READ-statement and DATA-statement. Three string constants ("ASSIGNING", "STRING", and "CONSTANTS") are assigned to three string variables (A$, B$, and C$). These assignments are then checked. On output there should be an error message for any faulty assignment. Each of these error messages should appear as follows: READ ASSIGNMENT FOR VARIABLE NUMBER <number>, FAILED. If there are no faulty assignments, then the following message should be printed: READ/DATA ASSIGNMENTS FOR STRING VARS., PASSED TEST.

74.3 READ/DATA for Numerical and String Variables Together

The object of this test is to verify that there should be a single sequence of data items, rather than separate sequences, for string data and for numeric data. However, if the DATA-list is a mixture of numeric constants and string constants, then the order of the mixture of numeric variables and string variables in the READ-list must correspond to the mixture in the DATA-list. That is, the type of a datum in the data sequence must correspond to the type of the variable to which it is to be assigned, (which means numeric variables require numeric constants as data and string variables require quoted strings or unquoted strings as data. On output there should be an error message for any incorrect assignment. Each of the error messages should appear as follows: READ ASSIGNMENT FOR VARIABLE NUMBER <number>, FAILED. If all assignments are correct then the following message should be printed: READ/DATA ASSIGNMENTS FOR NUM/STRG VARS. TOGETHER, PASSED.

74.4 Evaluation of Subscripted Variables

This test confirms (1) that subscripted expressions in the variable list are evaluated after values have been assigned to the variables preceding them (that is, to the left of them) in the list, and (2) that any previous LET assignments for the index should be ignored. The test first assigns values

to the elements of the array A(I) by a LET-statement. After the values have been assigned to the array A(I), the index I is assigned a value by a LET-statement in order to select one of the elements of the array A(I). Then, another value is assigned to the index I by a READ-statement and a DATA-statement. At this point the index value assigned by the LET-statement should be nullified, and the index value for I assigned by the READ and DATA-statements should take precedence. This controls the element assigned to the subscripted variable A(I) in the READ-list. On output one of two possible messages should be printed. If the test fails, then the following message should be printed: EVALUATION OF SUBSCRIPT EXP. IN VARIABLE LISTS, FAILED. If the test is passed, then the following message should be printed: EVALUATION OF SUBSCRIPT EXP. IN VARIABLE LISTS, PASSED.

```
**********
* PROGRAM FILE 74 *
**********
```

```
0010 PRINT "PROGRAM FILE 74"
0020 PRINT
0030 PRINT
0040 PRINT
0070 PRINT
0080 PRINT "          SECTION 74.1: READ/DATA FOR NUMERIC VARIABLES."
0090 PRINT
0100 PRINT "                SECTION 74.1.1: FOR SIMPLE VARIABLES."
0110 PRINT
0120 PRINT "                                BEGIN TEST."
0130 PRINT
0140 LET K=0
0150 DATA 123456,-4.76567,1.111E33
0160 READ M,N,O
0170 LET K=K+1
0180 IF M=123456 THEN 200
0190 GOSUB 270
0200 LET K=K+1
0210 IF N=-4.76567 THEN 230
0220 GOSUB 270
0230 LET K=K+1
0240 IF O=1.111E33 THEN 290
0250 GOSUB 270
0260 GOTO 320
0270 PRINT TAB(12);"READ ASSIGNMENT FOR VARIABLE NUMBER";K;", FAILED."
0280 RETURN
0290 PRINT "READ/DATA ASSIGNMENTS FOR SIMPLE VARIABLES, PASSED TEST."
0300 PRINT
0310 PRINT "                                END TEST."
0320 PRINT
0330 PRINT "              SECTION 74.1.2: FOR SUBSCRIPTED VARIABLES."
0340 PRINT
```

```
0350 PRINT "             SECTION 74.1.2.1: AS ONE-DIMENSIONAL ARRAYS."
0360 PRINT
0370 PRINT "                                  BEGIN TEST."
0380 PRINT
0390 DIM A(15),B(15),C(15),D(15)
0400 LET F=0
0410 DATA -9,-1.75,-8,-.75,-7,.25,-6,1.25,-5,2.25
0420 FOR I=1 TO 15
0430 READ A(I),B(I)
0440 LET C(I)=I-10
0450 LET D(I)=I-2.75
0460 NEXT I
0470 DATA -4,3.25,-3,4.25,-2,5.25,-1,6.25,0,7.25
0480 LET K=0
0490 FOR I=1 TO 15
0500 LET K=K+1
0510 IF A(I)=C(I) THEN 530
0520 GOSUB 620
0530 LET K=K+1
0540 IF B(I)=D(I) THEN 560
0550 GOSUB 620
0560 NEXT I
0570 DATA 1,8.25,2,9.25,3,10.25,4,11.25,5,12.25
0580 IF F=0 THEN 650
0590 PRINT
0600 PRINT "               DO TO THE ABOVE ERROR(S), TEST FAILED."
0610 GOTO 680
0620 LET F=F+1
0630 PRINT TAB(12);"READ ASSIGNMENT FOR VARIABLE NUMBER";K;", FAILED."
0640 RETURN
0650 PRINT "READ/DATA ASSIGNMENTS FOR ONE-DIMENSIONAL ARRAYS, PASSED."
0660 PRINT
0670 PRINT "                                   END TEST."
0680 PRINT
0690 PRINT "            SECTION 74.1.2.2: AS TWO-DIMENSIONAL ARRAYS."
0700 PRINT
0710 PRINT "                                  BEGIN TEST."
0720 PRINT
0730 DIM M(2,15),N(2,15),X(2,15),Y(2,15)
0740 LET F=0
0750 DATA -1,-4.25,0,-3.25,1,-2.25,2,-1.25,3,-.25,4,.75
0760 DATA 5,1.75,6,2.75,7,3.75,8,4.75,9,5.75,10,6.75
0770 FOR I=1 TO 2
0780 FOR J=1 TO 15
0790 READ M(I,J),N(I,J)
0800 LET X(I,J)=J-2*I
0810 LET Y(I,J)=I*J-5.25
0820 NEXT J
0830 NEXT I
0840 DATA 11,7.75,12,8.75,13,9.75,-3,-3.25,-2,-1.25,-1,.75
0850 LET K=0
0860 FOR I=1 TO 2
0870 FOR J=1 TO 15
0880 LET K=K+1
0890 IF M(I,J)=X(I,J) THEN 910
0900 GOSUB 1020
0910 LET K=K+1
```

```
0920 IF N(I,J)=Y(I,J) THEN 940
0930 GOSUB 1020
0940 NEXT J
0950 NEXT I
0960 DATA 0,2.75,1,4.75,2,6.75,3,8.75,4,10.75,5,12.75
0970 DATA 6,14.75,7,16.75,8,18.75,9,20.75,10,22.75,11,24.75
0980 IF F=0 THEN 1050
0990 PRINT
1000 PRINT "               DO TO THE ABOVE ERRORS(S), TEST FAILED."
1010 GOTO 1080
1020 LET F=F+1
1030 PRINT TAB(12);"READ ASSIGNMENT FOR VARIABLE NUMBER";K;", FAILED."
1040 RETURN
1050 PRINT "READ/DATA ASSIGNMENTS FOR TWO-DIMENSIONAL ARRAYS, PASSED."
1060 PRINT
1070 PRINT "                                    END TEST."
1080 PRINT
1090 PRINT "           SECTION 74.2: READ/DATA FOR STRING VARIABLES."
1100 PRINT
1110 PRINT "                                  BEGIN TEST."
1120 PRINT
1130 LET K=0
1140 DATA "ASSIGNING",STRING,"CONSTANTS"
1150 READ A$,B$,C$
1160 LET K=K+1
1170 IF A$="ASSIGNING" THEN 1190
1180 GOSUB 1260
1190 LET K=K+1
1200 IF B$="STRING" THEN 1220
1210 GOSUB 1260
1220 LET K=K+1
1230 IF C$="CONSTANTS" THEN 1280
1240 GOSUB 1260
1250 GOTO 1310
1260 PRINT TAB(12);"READ ASSIGNMENT FOR VARIABLE NUMBER";K;", FAILED."
1270 RETURN
1280 PRINT "READ/DATA ASSIGNMENTS FOR STRING VARIABLES, PASSED TEST."
1290 PRINT
1300 PRINT "                                    END TEST."
1310 PRINT
1320 PRINT "SECTION 74.3: READ/DATA FOR NUM. AND STRG. VARS. TOGETHER."
1330 PRINT
1340 PRINT "                                  BEGIN TEST."
1350 PRINT
1360 LET K=0
1370 DATA MIXING,123456,"NUMBERS",-4.76567,AND,1.111E33
1380 DATA STRINGS,-.654321,"IN",DATA
1390 READ A$,B,C$,D,E$,F,G$,H,I$,J$
1400 LET K=K+1
1410 IF A$="MIXING" THEN 1430
1420 GOSUB 1710
1430 LET K=K+1
1440 IF B=123456 THEN 1460
1450 GOSUB 1710
1460 LET K=K+1
1470 IF C$="NUMBERS" THEN 1490
1480 GOSUB 1710
```

```
1490 LET K=K+1
1500 IF D=-4.76567 THEN 1520
1510 GOSUB 1710
1520 LET K=K+1
1530 IF E$="AND" THEN 1550
1540 GOSUB 1710
1550 LET K=K+1
1560 IF F=1.111E33 THEN 1580
1570 GOSUB 1710
1580 LET K=K+1
1590 IF G$="STRINGS" THEN 1610
1600 GOSUB 1710
1610 LET K=K+1
1620 IF H=-.654321 THEN 1640
1630 GOSUB 1710
1640 LET K=K+1
1650 IF I$="IN" THEN 1670
1660 GOSUB 1710
1670 LET K=K+1
1680 IF J$="DATA" THEN 1730
1690 GOSUB 1710
1700 GOTO 1760
1710 PRINT TAB(12);"READ ASSIGNMENT FOR VARIABLE NUMBER";K;", FAILED."
1720 RETURN
1730 PRINT "READ/DATA ASSIGNMENTS FOR NUM/STRG VARS. TOGETHER, PASSED."
1740 PRINT
1750 PRINT "                                    END TEST."
1760 PRINT
1770 PRINT "          SECTION 74.4: EVALUATION OF SUBSCRIPTED VARIABLES."
1780 PRINT
1790 PRINT "                                  BEGIN TEST."
1800 PRINT
1810 DIM V(10)
1820 FOR I=1 TO 10
1830 LET V(I)=I-4*I^2
1840 NEXT I
1850 LET I=8
1860 DATA 6,3E-33
1870 READ I,V(I)
1880 IF V(8)<>-248 THEN 1900
1890 IF V(6)=3E-33 THEN 1920
1900 PRINT " EVALUATION OF SUBSCRIPT EXPS. IN VARIABLE LIST, FAILED."
1910 GOTO 1950
1920 PRINT " EVALUATION OF SUBSCRIPT EXPS. IN VARIABLE LIST, PASSED."
1930 PRINT
1940 PRINT "                                    END TEST."
1950 PRINT
1960 PRINT
1970 END
```

```
*****************
* SAMPLE OUTPUT *
*****************
```

PROGRAM FILE 74

SECTION 74.1: READ/DATA FOR NUMERIC VARIABLES.

SECTION 74.1.1: FOR SIMPLE VARIABLES.

BEGIN TEST.

READ/DATA ASSIGNMENTS FOR SIMPLE VARIABLES, PASSED TEST.

END TEST.

SECTION 74.1.2: FOR SUBSCRIPTED VARIABLES.

SECTION 74.1.2.1: AS ONE-DIMENSIONAL ARRAYS.

BEGIN TEST.

READ/DATA ASSIGNMENTS FOR ONE-DIMENSIONAL ARRAYS, PASSED.

END TEST.

SECTION 74.1.2.2: AS TWO-DIMENSIONAL ARRAYS.

BEGIN TEST.

READ/DATA ASSIGNMENTS FOR TWO-DIMENSIONAL ARRAYS, PASSED.

END TEST.

SECTION 74.2: READ/DATA FOR STRING VARIABLES.

BEGIN TEST.

READ/DATA ASSIGNMENTS FOR STRING VARIABLES, PASSED TEST.

END TEST.

SECTION 74.3: READ/DATA FOR NUM. AND STRG. VARS. TOGETHER.

BEGIN TEST.

READ/DATA ASSIGNMENTS FOR NUM/STRG VARS. TOGETHER, PASSED.

END TEST.

SECTION 74.4: EVALUATION OF SUBSCRIPTED VARIABLES.

BEGIN TEST.

EVALUATION OF SUBSCRIPT EXPS. IN VARIABLE LIST, PASSED.

END TEST.

75.0 EXCEPTION TEST - READ-STATEMENT ENCOUNTERS INSUFFICIENT DATA

The objective of this test is to verify that the implementation will diagnose whether a variable list in a READ-statement requires more data than is present in the remainder of the data sequence and reports this as an exception. The test has a DATA-statement at line 380 which has a DATA-list that is not in one-to-one correspondence with the variable list for the READ-statement at line 390. On output, there should appear some implementation-defined diagnostic reporting the error. However, should the implementation fail to recognize the error, the test has a message that reports test failure. The reader is referred to section 14.5 of BSR X3.60.

```
********************
* PROGRAM FILE 75 *
********************
```

```
0010 PRINT "PROGRAM FILE 75"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "        SECTION 75.0: FATAL ERROR CHECK ON READ-STATEMENT."
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0140 PRINT "THIS SYSTEM RECOGNIZES THE FOLLOWING AS FATAL ERRORS (THAT"
0150 PRINT "IS, SUSPENDING PROGRAM EXECUTION PENDING USER-DIRECTED RE-"
0160 PRINT "START PROCEDURES):"
0170 PRINT
0180 PRINT "      (1) WHEN THE VARIABLE LIST IN A READ-STATEMENT"
0190 PRINT "          REQUIRES MORE DATA THAN ARE PRESENT IN THE RE-"
0200 PRINT "          MAINDER OF THE DATA SEQUENCE."
0210 PRINT "      (2) WHEN A STRING DATUM DOES NOT MATCH THE TYPE OF THE"
0220 PRINT "          NUMERIC VARIABLE TO WHICH IT IS TO BE ASSIGNED."
0230 PRINT "      (3) WHEN THE CONVERSION OF A STRING DATUM CAUSES A"
0240 PRINT "          STRING OVERFLOW."
0250 PRINT
0260 PRINT "IF ALL OF THE ABOVE REFERRALS ARE CONSIDERED FATAL ERRORS,"
0270 PRINT "THEN THE TEST WILL HAVE PASSED."
0280 PRINT
0290 PRINT
0300 PRINT
0330 PRINT "                  (INSUFFICIENT DATA FOR VARIABLE LIST.)"
0340 PRINT
0350 PRINT
0360 PRINT "                                BEGIN TEST."
```

```
0370 PRINT
0380 DATA -2,16
0390 READ A,B,C
0400 PRINT "     IF THERE IS A PRINTOUT OF NUMBERS AFTER THIS STATEMENT"
0410 PRINT "THEN THE TEST WILL HAVE FAILED."
0420 PRINT A,B,C
0430 PRINT
0440 PRINT "                                   END TEST."
0450 PRINT
0460 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 75

SECTION 75.0: FATAL ERROR CHECK ON READ-STATEMENT.

THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER THIS SYSTEM RECOGNIZES THE FOLLOWING AS FATAL ERRORS (THAT IS, SUSPENDING PROGRAM EXECUTION PENDING USER-DIRECTED RE-START PROCEDURES):

(1) WHEN THE VARIABLE LIST IN A READ-STATEMENT REQUIRES MORE DATA THAN ARE PRESENT IN THE RE-MAINDER OF THE DATA SEQUENCE.
(2) WHEN A STRING DATUM DOES NOT MATCH THE TYPE OF THE NUMERIC VARIABLE TO WHICH IT IS TO BE ASSIGNED.
(3) WHEN THE CONVERSION OF A STRING DATUM CAUSES A STRING OVERFLOW.

IF ALL OF THE ABOVE REFERRALS ARE CONSIDERED FATAL ERRORS, THEN THE TEST WILL HAVE PASSED.

(INSUFFICIENT DATA FOR VARIABLE LIST.)

BEGIN TEST.

? OUT OF DATA IN LINE 390

76.0 EXCEPTION TEST - NON-MATCHING STRING DATUM ASSIGNED TO A NUMERIC VARIABLE

The objective of this test is to determine whether the implementation will recognize the attempt to read a string constant in the data sequence by a numeric variable as an exception. The test attempts to read a quoted string in a DATA-list by a simple numeric variable in a READ-list. On output, there should be some form of implementation-defined diagnostic. However, should the implementation fail to recognize the error, the test has a message printed that tells the user that the implementation failed the test. The reader is referred to section 14.5 of BSR X3.60.

****
* PROGRAM FILE 76 *
****

```
0010 PRINT "PROGRAM FILE 76"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 76.0"
0100 PRINT
0110 PRINT "(NON-MATCHING STRING DATUM ASSIGNED TO A NUMERIC VARIABLE.)"
0120 PRINT
0130 PRINT
0140 PRINT "                                BEGIN TEST."
0150 PRINT
0160 DATA "SIX"
0170 READ A
0180 PRINT "     EXECUTION OF PROGRAM WAS NOT SUSPENDED, THEREFORE,"
0190 PRINT "THE SYSTEM HAS FAILED THE TEST."
0200 PRINT
0210 PRINT "                                 END TEST."
0220 PRINT
0230 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 76

SECTION 76.0

(NON-MATCHING STRING DATUM ASSIGNED TO A NUMERIC VARIABLE.)

BEGIN TEST.

? VARIABLE IN LINE 170 INCOMPATIBLE WITH DATA

77.0 EXCEPTION TEST - ATTEMPTING A STRING DATUM OVERFLOW

This test determines at what point, if possible, the implementation recognizes the assigning of a string, consisting of more than 18 characters as an exception. Processors may accept the assigning of long strings. The point to this test is that when a processor cannot assign strings longer than 18 characters, then some form of diagnostic is required with suspension of program. This test has strings of various lengths assigned by the READ/DATA relationship. In fact, strings of lengths 19, 20, 30, 40, 50, and 58 characters are used. On output, either the strings are properly assigned or there should be some form of implementation-defined diagnostic reporting a string overflow error. The reader is referred to section 14.5 of BSR X3.60.

```
********************
* PROGRAM FILE 77 *
********************
```

```
0010 PRINT "PROGRAM FILE 77"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 77.0"
0100 PRINT
0110 PRINT "                            (A STRING OVERFLOW.)"
0120 PRINT
0130 PRINT
0140 PRINT "                                 BEGIN TEST."
0150 PRINT
0160 DATA "*********19*********"
0170 DATA "*********20*********"
0180 DATA "***************30***************"
0190 DATA "********************40********************"
0200 DATA "************************50************************"
0210 DATA "******************************58******************************"
0220 READ A$,B$,C$,D$,E$,F$
0230 PRINT "     IF THERE IS A PRINTOUT BELOW THIS PARAGRAPH AND NOT AN"
0240 PRINT "INDICATION OF A FATAL ERROR, THEN THIS SYSTEM SATISFIES"
0250 PRINT "MORE IN THIS RESPECT THAN IS REQUIRED BY MINIMAL BASIC."
0260 PRINT "THE NUMBERS TOWARD THE CENTER OR IN THE CENTER OF THE AS-"
0270 PRINT "TERISKS SIGNIFY THE LENGTH OF THE CHARACTER STRINGS ASSO-"
0280 PRINT "CIATED WITH EACH ASSIGNED STRING VARIABLE."
0290 PRINT
0300 PRINT A$
0310 PRINT B$
0320 PRINT C$
0330 PRINT D$
0340 PRINT E$
```

```
0350 PRINT F$
0360 PRINT
0370 PRINT "                                        END TEST."
0380 PRINT
0390 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 77

SECTION 77.0

(A STRING OVERFLOW.)

BEGIN TEST.

IF THERE IS A PRINTOUT BELOW THIS PARAGRAPH AND NOT AN INDICATION OF A FATAL ERROR, THEN THIS SYSTEM SATISFIES MORE IN THIS RESPECT THAN IS REQUIRED BY MINIMAL BASIC. THE NUMBERS TOWARD THE CENTER OR IN THE CENTER OF THE ASTERISKS SIGNIFY THE LENGTH OF THE CHARACTER STRINGS ASSOCIATED WITH EACH ASSIGNED STRING VARIABLE.

```
*********19********
**********20*********
***************30***************
********************40********************
************************50************************
******************************58******************************
```

END TEST.

78.0 SEMANTIC INTERPRETATION - A NUMERIC VALUE IN A DATA LIST CAUSES AN UNDERFLOW

This test verifies that the implementation recognizes the semantic interpretation required when a positive numerical constant, which is too small to be represented by the machine, is assigned. The READ/DATA relationship should ignore the value being assigned by making an assignment of zero to the value and continuing the program. The test has a DATA-statement which lists a numerical value of the magnitude 9.0E-99999 at line 360. On output, there should be a message which should tell the user to look for a zero as a printout following that message. If zero is printed, then the implementation will have passed the test. The reader is referred to section 14.4 in BSR X3.60.

```
********************
* PROGRAM FILE 78 *
********************
```

```
0010 PRINT "PROGRAM FILE 78"
0020 PRINT
0030 PRINT
0040 PRINT
0260 PRINT
0270 PRINT
0280 PRINT
0290 PRINT "                              SECTION 78.0"
0300 PRINT
0310 PRINT "                  (A NUMERIC DATUM CAUSES AN UNDERFLOW.)"
0320 PRINT
0330 PRINT
0340 PRINT "                                 BEGIN TEST."
0350 PRINT
0360 DATA 9.0E-99999
0370 READ A
0380 PRINT "     IF THE NUMBER PRINTED AFTER THIS STATEMENT IS ZERO,"
0390 PRINT "THEN THE SYSTEM WILL HAVE PASSED THE TEST."
0400 PRINT A
0410 PRINT
0420 PRINT "                                   END TEST."
0430 PRINT
0440 END
```

```
******************
* SAMPLE OUTPUT *
```

****

PROGRAM FILE 78

SECTION 78.0

(A NUMERIC DATUM CAUSES AN UNDERFLOW.)

BEGIN TEST.

IF THE NUMBER PRINTED AFTER THIS STATEMENT IS ZERO,
THEN THE SYSTEM WILL HAVE PASSED THE TEST.
0

END TEST.

79.0 EXCEPTION TEST - A NUMERIC VALUE IN A DATA STATEMENT CAUSES AN OVERFLOW

This test verifies that the implementation will recognize the assigning, by the READ/DATA relationship, of a number that causes overflow, as an exception. The exception recovery procedure should cause the implementation-defined machine infinity to be assigned instead. The test has a DATA-statement in line 170 which lists a numerical value of magnitude 9.99999E99999. On output, there should be a message telling the user to look for the implementation-defined machine infinity as a printout following the message. If a positive machine infinity is printed, then the implementation will have passed the test. The reader is referred to section 14.5 of BSR X3.60.

****
* PROGRAM FILE 79 *
****

```
0010 PRINT "PROGRAM FILE 79"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                                SECTION 79.0"
0100 PRINT
0110 PRINT "     (A NUMERIC DATUM CAUSES AN OVERFLOW, POSITIVE MACHINE"
0120 PRINT "                                  INFINITY.)"
0130 PRINT
0140 PRINT
0150 PRINT "                                  BEGIN TEST."
0160 PRINT
0170 DATA 9.99999E99999
0180 READ A
0190 PRINT "     IF THE NUMBER PRINTED BELOW THIS STATEMENT IS POSITIVE"
0200 PRINT "AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST"
0210 PRINT "WILL HAVE PASSED ON THIS SYSTEM."
0220 PRINT A
0230 PRINT
0240 PRINT "                                   END TEST."
0250 PRINT
0260 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 79

SECTION 79.0

(A NUMERIC DATUM CAUSES AN OVERFLOW, POSITIVE MACHINE INFINITY.)

BEGIN TEST.

?OVERFLOW IN LINE 180

IF THE NUMBER PRINTED BELOW THIS STATEMENT IS POSITIVE AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST WILL HAVE PASSED ON THIS SYSTEM.
1.70141E+38

END TEST.

80.0 EXCEPTION TEST - OVERFLOW CAUSED BY A NUMERIC VALUE IN A DATA-STATEMENT (CONTINUED)

This test verifies that the implementation will assign negative machine infinity when a negative numeric datum causes an overflow. There is a DATA-statement which has the numerical constant -9.99999E99999 in line 150. On output, a message should be printed telling the user to look for negative machine infinity. If that value is printed following the message, then implementation will have passed the test. The reader is referred to section 14.5 of BSR X3.60 .

****
* PROGRAM FILE 80 *
****

```
0010 PRINT "PROGRAM FILE 80"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "   SECTION 80.0:  OVERFLOW CAUSED BY A NUMERIC DATUM."
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "                                BEGIN TEST."
0140 PRINT
0150 DATA -9.99999E99999
0160 READ A
0170 PRINT "     IF THE NUMBER PRINTED BELOW THIS STATEMENT IS NEGATIVE"
0180 PRINT "AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST"
0190 PRINT "WILL HAVE PASSED ON THIS SYSTEM."
0200 PRINT A
0210 PRINT
0220 PRINT "                                  END TEST."
0230 PRINT
0240 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 80

SECTION 80.0: OVERFLOW CAUSED BY A NUMERIC DATUM.

BEGIN TEST.

?OVERFLOW IN LINE 160

IF THE NUMBER PRINTED BELOW THIS STATEMENT IS NEGATIVE
AND THE MACHINE INFINITY FOR THIS SYSTEM, THEN THE TEST
WILL HAVE PASSED ON THIS SYSTEM.
-1.70141E+38

END TEST.

81.0 RESTORING READ DATA

This test verifies that, through the use of the RESTORE-statement, data from the data sequence can be reread. The test on the RESTORE-statement is accomplished by first reading each datum of the DATA-list and then assigning each value read to a variable. These assignments are then checked. If any faulty assignments are made, then an error message should be printed for each incorrect assignment. Each of these error messages should be printed as follows: READ ASSIGNMENT FOR VARIABLE NUMBER <number>, FAILED. If all assignments were correct, then, using the RESTORE-statement, a second set of variables should be assigned the same values as the first. The correctness of the assignments are then checked by IF-THEN comparisons between the values of the first and the second set of variables. If there are any incorrect comparisons, then there should be an error message for each incorrect reassigned value. Each of the error messages should be printed as follows: REREAD ASSIGNMENT FOR VARIABLE NUMBER <number>, FAILED. If all assignments in the rereading process were correct then the following message should be printed: RESTORING DATA TO BE REREAD FOR ASSIGNMENT, PASSED. The reader is referred to section 14.4 of BSR X3.60.

****
* PROGRAM FILE 81 *
****

```
0010 PRINT "PROGRAM FILE 81"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "     CAUTION:  THE NATURE OF THE RESTORE-STATEMENT DOES"
0100 PRINT "NOT ALLOW THIS SECTION TO BE TESTED WITH OTHER SECTIONS"
0110 PRINT "USING A DATA-STATEMENT."
0120 PRINT
0130 PRINT "                    SECTION 81.0: RESTORING READ DATA."
0140 PRINT
0150 PRINT "                                  BEGIN TEST."
0160 PRINT
0170 LET F=0
0180 LET K=0
0190 DATA "RESTORING",1.23E-9,READ,-9.876E22,"DATA"
0200 READ A$,B,C$,D,E$
0210 LET K=K+1
0220 IF A$="RESTORING" THEN 240
0230 GOSUB 570
0240 LET K=K+1
0250 IF B=1.23E-9 THEN 270
0260 GOSUB 570
0270 LET K=K+1
0280 IF C$="READ" THEN 300
```

```
0290 GOSUB 570
0300 LET K=K+1
0310 IF D=-9.876E22 THEN 330
0320 GOSUB 570
0330 LET K=K+1
0340 IF E$="DATA" THEN 370
0350 GOSUB 570
0360 GOTO 600
0370 IF F<>0 THEN 600
0380 LET K=0
0390 RESTORE
0400 READ M$,N,O$,P,Q$
0410 LET K=K+1
0420 IF M$=A$ THEN 440
0430 GOSUB 630
0440 LET K=K+1
0450 IF N=B THEN 470
0460 GOSUB 630
0470 LET K=K+1
0480 IF O$=C$ THEN 500
0490 GOSUB 630
0500 LET K=K+1
0510 IF P=D THEN 530
0520 GOSUB 630
0530 LET K=K+1
0540 IF Q$=E$ THEN 650
0550 GOSUB 630
0560 GOTO 680
0570 LET F=G+1
0580 PRINT TAB(12),"READ ASSIGNMENT FOR VARIABLE NUMBER";K;", FAILED."
0590 RETURN
0600 PRINT
0610 PRINT "               THEREFORE, RESTORE TEST WILL NOT CONTINUE."
0620 GOTO 680
0630 PRINT TAB(12),"REREAD ASSIGNMENT FOR VARIABLE NO.";K;", FAILED."
0640 RETURN
0650 PRINT "RESTORING READ DATA TO BE REREAD FOR ASSIGNMENT, PASSED."
0660 PRINT
0670 PRINT "                                   END TEST."
0680 PRINT
0690 PRINT
0700 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 81

CAUTION: THE NATURE OF THE RESTORE STATEMENT DOES NOT ALLOW THIS SECTION TO BE TESTED WITH OTHER SECTIONS USING A DATA-STATEMENT.

SECTION 81.0: RESTORING READ DATA.

BEGIN TEST.

RESTORING READ DATA TO BE REREAD FOR ASSIGNMENT, PASSED.

END TEST.

82.0 INPUT STATEMENT FOR NUMERIC CONSTANTS

The next several test sections emphasize user interaction with a running program. This is accomplished through the use of the INPUT-statement (see section 13 of BSR X3.60). By using the INPUT-statement, data can be entered as quoted strings, unquoted strings, numeric constants or a mixture of all three types. However, upon call for data by an input-prompt, there is a restriction on the order in which data is supplied in the input-reply. That is, the type of each datum in the input-reply must correspond to the type of the variable to which it is to be assigned (numeric constants must be supplied as input for numeric variables, and either quoted strings or unquoted strings must be supplied as input for string variables). If the response to input for a string variable is an unquoted string, leading and trailing spaces are to be ignored. Subscript expressions in the variable list should be evaluated after values have been assigned to the variables preceding them, that is, from left to right in the variable list.

The standard specifies that in batch mode, input-reply is requested from an external source by an implementation-defined means. If these tests are run in batch mode, then the user will have to use the appropriate external source and program the proper data input form before these codes can be executed.

The objective of the program in this section, specifically, is to determine whether the implementation recognizes the assigning of numerical constants by the use of the INPUT-statement.

82.1 Input of a Numeric Constant

The objective of this exercise is to test simple interaction by requesting the user to input a single value at a time. The test requests that all three of the numeric constant forms (NR1, NR2, and NR3 forms) be individually entered by the user. Each prompt message (which should be followed by an input-prompt) tells the user what number, and in what form to enter the number for a proper response. To allow for possible data input errors during input-reply responses, each datum is checked by the test for proper format. If an incorrect value is found to have been entered, the user will be given only two more possible chances to correct his error. If the input is not correct after that, the following message will be printed: FAILURE TO ENTER PROPER DATA. After this message, the test will stop. If all of the proper data is entered, a comparative output should be printed. There are four columns of output. The first specifies the standard value, the second an option, if any, the third reports the system value after input/output conversion, and the final column reports any internal relative error.

82.2 Input of Numeric Constants As a Line of Data Separated by Commas

The objective of this test is again to determine whether numeric constants (in either of the three forms NR1, NR2, NR3, or all three) can be assigned by user interaction. In this test, however, the user is requested to enter several numbers separated by commas rather than a single number as above. Except for the input-prompts requiring several numbers rather than a

single number, the structure of this test and output is similar to section 82.1.

```
                    ********************
                    * PROGRAM FILE 82 *
                    ********************
```

```
0010 PRINT "PROGRAM FILE 82"
0020 PRINT
0030 PRINT
0040 PRINT
0080 PRINT "                SECTION 82.0: INPUT OF NUMERIC CONSTANTS."
0090 PRINT
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "                              SECTION 82.1"
0140 PRINT
0150 PRINT "                        (AS A SINGLE DATUM VALUE.)"
0160 PRINT
0170 PRINT
0180 PRINT "                                BEGIN TEST."
0190 PRINT
0200 DIM E(8),N(8),D(8),Z(8)
0202 DATA 8.13008E-7, 8.10045E-7, 8.10005E-7, 1E-7
0203 DATA 1.52830E-7, 1.26727E-7, 1.01256E-7, 8.10045E-7
0204 FOR I = 1 TO 8
0205 READ Z(I)
0206 NEXT I
0210 LET I=1
0220 LET C=123
0230 LET A$="123"
0240 GOSUB 1050
0250 LET C=-12345
0260 LET A$="-12345"
0270 GOSUB 1050
0280 LET C=12345.6
0290 LET A$="12345.6"
0300 GOSUB 1050
0310 LET C=-99999.9
0320 LET A$="-99999.9"
0330 GOSUB 1050
0340 LET C=6.54321E-20
0350 LET A$="6.54321E-20"
0360 GOSUB 1050
0370 LET C=-7.891E25
0380 LET A$="-7.891E25"
0390 GOSUB 1050
0400 LET C=-987.6E-27
```

```
0410 LET A$="-987.6E-27"
0420 GOSUB 1050
0430 LET C=12345E18
0440 LET A$="12345E18"
0450 GOSUB 1050
0460 PRINT
0470 PRINT
0480 PRINT "     IF THE RELATIVE ERROR IS OUT OF TOLERANCE,"
0490 PRINT "THEN AN ASTERISK WILL FOLLOW THAT RELATIVE"
0500 PRINT "ERROR SIGNIFYING ONE OF TWO POSSIBILITIES:"
0510 PRINT
0520 PRINT "     (1) USER INPUT ERROR(S)."
0530 PRINT "     (2) FAILURE OF THE SYSTEM TO MAINTAIN SIX SIGNIFICANT"
0540 PRINT "DIGITS OF PRECISION."
0550 PRINT
0560 PRINT "OTHERWISE, NO ASTERISKS MEAN TEST PASSED."
0570 PRINT
0580 PRINT
0590 PRINT "STANDARD","OPTIONAL","SYSTEM","RELATIVE"
0600 PRINT " OUTPUT "," OUTPUT ","OUTPUT"," ERROR  "
0610 PRINT
0620 FOR I=1 TO 8
0630 ON I GOTO 640,690,740,790,830,880,930,980
0640 IF N(I)=0 THEN 670
0650 PRINT " 123 ","NONE",D(I),E(I);"*"
0660 GOTO 1020
0670 PRINT " 123 ","NONE",D(I),E(I)
0680 GOTO 1020
0690 IF N(I)=0 THEN 720
0700 PRINT "-12345 ","NONE",D(I),E(I);"*"
0710 GOTO 1020
0720 PRINT "-12345 ","NONE",D(I),E(I)
0730 GOTO 1020
0740 IF N(I)=0 THEN 770
0750 PRINT " 12345.6 ","NONE",D(I),E(I);"*"
0760 GOTO 1020
0770 PRINT " 12345.6 ","NONE",D(I),E(I)
0780 GOTO 1020
0790 IF N(I)=0 THEN 810
0800 GOTO 1020
0810 PRINT "-99999.9 ","NONE",D(I),E(I)
0820 GOTO 1020
0830 IF N(I)=0 THEN 860
0840 PRINT " 6.54321E-20 ","NONE",D(I),E(I);"*"
0850 GOTO 1020
0860 PRINT " 6.54321E-20 ","NONE",D(I),E(I)
0870 GOTO 1020
0880 IF N(I)=0 THEN 910
0890 PRINT "-7.89100E25 ","-7.89100E+25 ",D(I),E(I);"*"
0900 GOTO 1020
0910 PRINT "-7.89100E25 ","-7.89100E+25 ",D(I),E(I)
0920 GOTO 1020
0930 IF N(I)=0 THEN 960
0940 PRINT "-9.87600E-25 ","NONE",D(I),E(I);"*"
0950 GOTO 1020
0960 PRINT "-9.87600E-25 ","NONE",D(I),E(I)
0970 GOTO 1020
```

```
0980 IF N(I)=0 THEN 1010
0990 PRINT " 1.2345E22 "," 1.2345E+22 ",D(I),E(I);"*"
1000 GOTO 1020
1010 PRINT " 1.2345E22 "," 1.2345E+22 ",D(I),E(I)
1020 NEXT I
1030 PRINT
1040 GOTO 1300
1050 LET N(I)=0
1060 PRINT "ENTER THE FOLLOWING NUMERAL AS IS: ";A$
1070 INPUT D(I)
1080 IF C<>D(I) THEN 1120
1090 LET E(I)=(C-D(I))/C
1100 LET I=I+1
1110 RETURN
1120 IF ABS(C-D(I))>ABS(C)*Z(I) THEN 1160
1130 LET E(I)=(C-D(I))/C
1140 LET I=I+1
1150 RETURN
1160 FOR J=1 TO 2
1170 PRINT "     POSSIBLE INPUT ERROR, PLEASE RE-ENTER DATA IN THE"
1180 PRINT "FOLLOWING FORM: ";A$
1190 INPUT D(I)
1200 IF C=D(I) THEN 1090
1210 IF ABS(C-D(I))>ABS(C)*Z(I) THEN 1250
1220 LET E(I)=(C-D(I))/C
1230 LET I=I+1
1240 RETURN
1250 NEXT J
1260 N(I)=1
1270 LET E(I)=(C-D(I))/C
1280 LET I=I+1
1290 RETURN
1300 PRINT
1310 PRINT
1320 PRINT
1330 PRINT "                                SECTION 82.2"
1340 PRINT
1350 PRINT "               (AS A LINE OF DATA SEPARATED BY COMMAS.)"
1360 PRINT
1370 PRINT
1380 PRINT "                                  BEGIN TEST."
1390 PRINT
1400 DIM B(3),A(3),R(3),T(3),H(3)
1401 DATA 8.10373E-7, 8.10005E-7, 1.01266E-7
1402 FOR I = 1 TO 3
1403 READ H(I)
1404 NEXT I
1410 LET B(1)=1234
1420 LET B(2)=123.456
1430 LET B(3)=-98.76E21
1440 PRINT "     PLEASE ENTER THE FOLLOWING LIST OF NUMERALS IN THE EX-"
1450 PRINT "ACT ORDER WHICH FOLLOWS: 1234,123.456,-98.76E21"
1460 INPUT A(1),A(2),A(3)
1470 LET F=0
1480 FOR I=1 TO 3
1490 IF B(I)<>A(I) THEN 1530
1500 LET R(I)=(B(I)-A(I))/B(I)
```

```
1510 LET T(I)=0
1520 GOTO 1600
1530 IF ABS(B(I)-A(I))>ABS(B(I))*H(I) THEN 1570
1540 LET R(I)=(B(I)-A(I))/B(I)
1550 LET T(I)=0
1560 GOTO 1600
1570 LET R(I)=(B(I)-A(I))/B(I)
1580 LET T(I)=1
1590 LET F=1
1600 NEXT I
1610 IF F=0 THEN 1860
1620 FOR J=1 TO 2
1630 PRINT "     POSSIBLE INPUT ERROR, PLEASE RE-ENTER THE FOLLOWING"
1640 PRINT "LIST EXACTLY AS ORDERED: 1234,123.456,-98.76E21"
1650 INPUT A(1),A(2),A(3)
1660 GOSUB 1710
1670 IF F<>0 THEN 1690
1680 GOTO 1860
1690 NEXT J
1700 GOTO 1860
1710 LET F=0
1720 FOR I=1 TO 3
1730 IF B(I)<>A(I) THEN 1770
1740 LET R(I)=(B(I)-A(I))/B(I)
1750 LET T(I)=0
1760 GOTO 1840
1770 IF ABS(B(I)-A(I))>ABS(B(I))*H(I) THEN 1810
1780 LET R(I)=(B(I)-A(I))/B(I)
1790 LET T(I)=0
1800 GOTO 1840
1810 LET R(I)=(B(I)-A(I))/B(I)
1820 LET T(I)=1
1830 LET F=1
1840 NEXT I
1850 RETURN
1860 PRINT
1870 PRINT
1880 PRINT "     IF THE RELATIVE ERROR IS OUT OF TOLERANCE,"
1890 PRINT "THEN AN ASTERISK WILL FOLLOW THAT RELATIVE"
1900 PRINT "ERROR SIGNIFYING ONE OF TWO POSSIBILITIES:"
1910 PRINT
1920 PRINT "     (1) USER INPUT ERROR(S)."
1930 PRINT "     (2) FAILURE OF THE SYSTEM TO MAINTAIN SIX SIGNIFICANT"
1940 PRINT "DIGITS OF PRECISION."
1950 PRINT
1960 PRINT "OTHERWISE, NO ASTERISKS MEAN TEST PASSED."
1970 PRINT
1980 PRINT
1990 PRINT "STANDARD","OPTIONAL","SYSTEM","RELATIVE"
2000 PRINT " OUTPUT "," OUTPUT ","OUTPUT"," ERROR  "
2010 PRINT
2020 FOR I=1 TO 3
2030 ON I GOTO 2040,2090,2140
2040 IF T(I)=0 THEN 2070
2050 PRINT " 1234 ","NONE",A(I),R(I);"*"
2060 GOTO 2180
2070 PRINT " 1234 ","NONE",A(I),R(I)
```

```
2080 GOTO 2180
2090 IF T(I)=0 THEN 2120
2100 PRINT " 123.456 ","NONE",A(I),R(I);"*"
2110 GOTO 2180
2120 PRINT " 123.456 ","NONE",A(I),R(I)
2130 GOTO 2180
2140 IF T(I)=0 THEN 2170
2150 PRINT "-9.87600E22 ","-9.87600E+22 ",A(I),R(I);"*"
2160 GOTO 2180
2170 PRINT "-9.87600E22 ","-9.87600E+22 ",A(I),R(I)
2180 NEXT I
2190 PRINT
2200 PRINT
2210 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

```
PROGRAM FILE 82

               SECTION 82.0: INPUT OF NUMERIC CONSTANTS.

                          SECTION 82.1

                    (AS A SINGLE DATUM VALUE.)

                            BEGIN TEST.

ENTER THE FOLLOWING NUMERAL AS IS: 123
?123
ENTER THE FOLLOWING NUMERAL AS IS: -12345
?-12345
ENTER THE FOLLOWING NUMERAL AS IS: 12345.6
?12345.6
ENTER THE FOLLOWING NUMERAL AS IS: -99999.9
?-99999.9
ENTER THE FOLLOWING NUMERAL AS IS: 6.54321E-20
?6.54321E-20
ENTER THE FOLLOWING NUMERAL AS IS: -7.891E25
?-7.891E25
ENTER THE FOLLOWING NUMERAL AS IS: -987.6E-27
?-987.6E-27
ENTER THE FOLLOWING NUMERAL AS IS: 12345E18
```

?12345E18

IF THE RELATIVE ERROR IS OUT OF TOLERANCE,
THEN AN ASTERISK WILL FOLLOW THAT RELATIVE
ERROR SIGNIFYING ONE OF TWO POSSIBILITIES:

(1) USER INPUT ERROR(S).
(2) FAILURE OF THE SYSTEM TO MAINTAIN SIX SIGNIFICANT
DIGITS OF PRECISION.

OTHERWISE, NO ASTERISKS MEAN TEST PASSED.

STANDARD OUTPUT	OPTIONAL OUTPUT	SYSTEM OUTPUT	RELATIVE ERROR
123	NONE	123	0
-12345	NONE	-12345	0
12345.6	NONE	12345.6	0
-99999.9	NONE	-99999.9	0
6.54321E-20	NONE	6.54321E-20	0
-7.89100E25	-7.89100E+25	-7.89100E+25	0
-9.87600E-25	NONE	-9.87600E-25	0
1.2345E22	1.2345E+22	1.23450E+22	0

SECTION 82.2

(AS A LINE OF DATA SEPARATED BY COMMAS.)

BEGIN TEST.

PLEASE ENTER THE FOLLOWING LIST OF NUMERALS IN THE EX-
ACT ORDER WHICH FOLLOWS: 1234,123.456,-98.76E21
?1234,123.456,-98.76E21

IF THE RELATIVE ERROR IS OUT OF TOLERANCE,
THEN AN ASTERISK WILL FOLLOW THAT RELATIVE
ERROR SIGNIFYING ONE OF TWO POSSIBILITIES:

(1) USER INPUT ERROR(S).
(2) FAILURE OF THE SYSTEM TO MAINTAIN SIX SIGNIFICANT
DIGITS OF PRECISION.

OTHERWISE, NO ASTERISKS MEAN TEST PASSED.

STANDARD OUTPUT	OPTIONAL OUTPUT	SYSTEM OUTPUT	RELATIVE ERROR
1234	NONE	1234	0
123.456	NONE	123.456	0
-9.87600E22	-9.87600E+22	-9.87600E+22	0

83.0 INPUT OF NUMERIC DATA TO SUBSCRIPTED VARIABLES AND UNQUOTED STRINGS

83.1 Evaluation of a Subscripted Variable in a Variable List

This test determines whether the implementation will allow a variable, used in a subscript, to be assigned a value in the same input list as the subscripted variable itself. The assignment of the value must of course be prior to the use of the variable in the subscript.

The test tells the user by input-prompt messages how to enter a list of ten given values for ten elements of an array A(I). After the proper input of the given list of values has been accomplished, the list will be printed out for user verification. After this printout, the user will be asked by an input-prompt message to choose one of the digits 1, 2, ..., 10 and then enter the chosen digit for the value of I in the variable list, and for the variable A(I) in the variable list, enter the value 54.8. After the values for I and A(I) have been entered properly, the elements of the array A(I) will again be printed out for comparison with the first printout of the elements of A(I). The comparison should show that the subscripted element which has the selected I value should now have an assigned value of 54.8.

83.2 Allowable Characters for Unquoted String Inputs

The objective of this test is to assign strings using the INPUT-statement in order to verify that the implementation allows the assigning of all of the characters specified by the standard for unquoted strings. These include all plain-string-characters and the character space (refer to the standard for the description of plain-string-characters). The test uses input-prompt messages to the user informing him which subset of plain string characters should be entered and the form or order in which each subset of characters should be entered for each response. All data are checked for proper form, and, if the data are entered incorrectly, the user will be given three chances to enter them correctly. If the data are still found to be incorrectly entered, the following message will be printed: FAILURE TO ENTER PROPER DATA. The test will then terminate. If the data are entered properly, a printout of the characters will be given. The reader is referred to sections 3 and 13 of BSR X3.60.

```
**********
* PROGRAM FILE 83 *
**********
```

```
0010 PRINT "PROGRAM FILE 83"
0060 PRINT
0070 PRINT
```

```
0080 PRINT
0090 PRINT "                              SECTION 83.1"
0100 PRINT
0110 PRINT "  EVALUATION OF A SUBSCRIPTED VARIABLE IN A VARIABLE LIST"
0130 PRINT
0140 PRINT
0150 PRINT "                                    BEGIN TEST."
0160 PRINT
0170 PRINT
0180 PRINT
0190 PRINT "     AFTER EACH SUCCESSIVE INPUT-PROMPT, ENTER SEQUENTIALLY"
0200 PRINT "ONE OF THE FOLLOWING LISTED NUMBERS:1.5,2.5,3.5,4.5,5.5,"
0210 PRINT "6.5,7.5,8.5,9.5,10.5"
0220 FOR I=1 TO 10
0230 INPUT A(I)
0240 GOSUB 540
0250 NEXT I
0260 PRINT
0270 PRINT "     LISTED BELOW ARE YOUR 10 INPUTTED DATA VALUES FOR THE"
0280 PRINT "ARRAY A(I).  THE ELEMENTS ARE LISTED IN THE ORDER OF A(1),"
0290 PRINT "A(2),A(3),...,A(10) FROM TOP TO BOTTOM RESPECTIVELY."
0300 PRINT
0310 FOR I=1 TO 10
0320 PRINT TAB(29),A(I)
0330 NEXT I
0340 PRINT
0350 PRINT "     AFTER THE INPUT-PROMPT, YOU ARE TO ENTER ONLY TWO IN-"
0360 PRINT "PUT-VALUES.  (1) FOR THE FIRST INPUT-VALUE, SELECT ONE OF"
0370 PRINT "THE DIGITS 1,2,3,...,10 AS THE SELECTION OF ONE OF THE ELE-"
0380 PRINT "MENTS OF ARRAY A(I).  (2) FOR THE SECOND INPUT-VALUE, ENTER"
0390 PRINT "THEN NUMBER 54.8."
0400 INPUT I,A(I)
0410 GOSUB 630
0420 PRINT
0430 PRINT "     LISTED BELOW ARE AGAIN THE ELEMENTS OF ARRAY A(I) IN"
0440 PRINT "THE SAME ORDER A(1),A(2),A(3),...,A(10) FROM TOP TO BOTTOM"
0450 PRINT "RESPECTIVELY, BUT IN THIS LIST THE VALUE OF YOUR SELECTED"
0460 PRINT "ELEMENT SHOULD BE 54.8 AND NOT THE VALUE IT HAD IN THE LIST"
0470 PRINT "ABOVE."
0480 PRINT
0490 FOR I=1 TO 10
0500 PRINT TAB(29),A(I)
0510 NEXT I
0520 PRINT
0530 GOSUB 780
0540 IF A(I)<>I+.5 THEN 560
0550 RETURN
0560 FOR K=1 TO 3
0570 PRINT "     DATA ERROR, PLEASE ENTER DATA AFTER THE INPUT-PROMPT"
0580 PRINT "IN THE FOLLOWING FORM:";I+.5
0590 INPUT A(I)
0600 IF A(I)=I+.5 THEN 550
0610 NEXT K
0620 GOTO 770
0630 IF I<1 THEN 670
0640 IF I>10 THEN 670
0650 IF A(I)<>54.8 THEN 670
```

```
0660 RETURN
0670 FOR J=1 TO 3
0680 PRINT "     DATA ERROR, RE-ENTER YOUR SELECTED DIGIT OF THE DIGITS"
0690 PRINT "1,2,3,...,10 AS THE FIRST DATA INPUT AND THE NUMBER 54.8 AS"
0700 PRINT "THE SECOND DATA ENTRY.  PLEASE MAKE ALL ENTRIES AFTER THE"
0710 PRINT "INPUT-PROMPT."
0720 INPUT I,A(I)
0730 IF I<1 THEN 760
0740 IF I>10 THEN 760
0750 IF A(I)=54.8 THEN 660
0760 NEXT J
0770 PRINT "                        FAILURE TO ENTER PROPER DATA."
0780 PRINT
0790 PRINT "                                  END TEST."
0820 PRINT
0830 PRINT
0840 PRINT
0850 PRINT
0860 PRINT "                            SECTION 83.2"
0870 PRINT
0880 PRINT "       ALLOWABLE CHARACTERS FOR UNQUOTED STRING INPUTS"
0890 PRINT
0900 PRINT
0910 PRINT "                                  BEGIN TEST."
0920 PRINT
0930 PRINT "     AT EACH SUCCESSIVE INPUT-PROMPT, ENTER ONE OF THE ROWS"
0940 PRINT "OF CHARACTERS BELOW AS THEY APPEAR FROM TO TO BOTTOM:"
0950 PRINT
0960 PRINT "ABCDEFGHIJKLM"
0970 PRINT "NOPQRSTUVWXYZ"
0980 PRINT "0123456789"
0990 PRINT "*():$=><"
1000 PRINT "- %;+.?/ "
1010 INPUT A$
1020 INPUT B$
1030 INPUT C$
1040 INPUT D$
1050 INPUT E$
1060 GOSUB 1380
1070 PRINT
1080 PRINT "     AFTER THE INPUT-PROMPT, ENTER THE DIGIT 1, SPACE OVER"
1090 PRINT "EIGHT SPACES, AND THEN TYPE THE DIGIT 1 AGAIN."
1100 INPUT F$
1110 GOSUB 1660
1120 PRINT
1130 PRINT
1140 PRINT "UNQUOTED-STRING-CHARACTER= SPACE/PLAIN-STRING-CHARACTER"
1150 PRINT
1160 PRINT "                           PLAIN-STRING-CHARACTERS."
1170 PRINT
1180 PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789*():$=><- %;+.?/"
1190 PRINT
1200 PRINT "     IF THE LINE OF CHARACTERS PRINTED BELOW, APPEAR IN THE"
1210 PRINT "SAME FORM AS THE LINE OF CHARACTERS ABOVE, CHECK TEST PASS-"
1220 PRINT "ED."
1230 PRINT
1240 PRINT A$;B$;C$;D$;E$
```

```
1250 PRINT
1260 PRINT "                              SPACE (IMBEDDED)."
1270 PRINT
1280 PRINT "     IF AFTER THE COLUMN NUMBERED HEADINGS, THERE ARE EIGHT"
1290 PRINT "COUNTABLE SPACES BETWEEN THE ONES- CHECK TEST PASSED."
1300 PRINT
1310 PRINT "00000000011111111112222222222333333333344444444445";
1320 PRINT "55555555566666666677 7"
1330 PRINT "12345678901234567890123456789012345678901234567890";
1340 PRINT "123456789012345678901"
1350 PRINT F$
1360 PRINT
1370 GOTO 1760
1380 IF A$<>"ABCDEFGHIJKLM" THEN 1440
1390 IF B$<>"NOPQRSTUVWXYZ" THEN 1440
1400 IF C$<>"0123456789" THEN 1440
1410 IF D$<>"*():$=><" THEN 1440
1420 IF E$<>"- %;+.?/" THEN 1440
1430 RETURN
1440 FOR I=1 TO 3
1450 PRINT "     DATA ERROR, PLEASE SUCCESSIVELY ENTER AFTER EACH INPUT"
1460 PRINT "PROMPT ONE OF THE LIST OF CHARACTERS AS THEY APPEAR FROM"
1470 PRINT "TOP TO BOTTOM."
1480 PRINT
1490 PRINT "ABCDEFGHIJKLM"
1500 PRINT "NOPQRSTUVWXYZ"
1510 PRINT "0123456789"
1520 PRINT "*():$=><"
1530 PRINT "- %;+.?/ "
1540 INPUT A$
1550 INPUT B$
1560 INPUT C$
1570 INPUT D$
1580 INPUT E$
1590 IF A$<>"ABCDEFGHIJKLM" THEN 1640
1600 IF B$<>"NOPQRSTUVWXYZ" THEN 1640
1610 IF C$<>"0123456789" THEN 1640
1620 IF D$<>"*():$=><" THEN 1640
1630 IF E$="- %;+.?/" THEN 1430
1640 NEXT I
1650 GOTO 1750
1660 IF F$<>"1         1" THEN 1680
1670 RETURN
1680 FOR I=1 TO 3
1690 PRINT "     DATA ERROR, PLEASE ENTER AFTER THE INPUT-PROMPT THE"
1700 PRINT "DIGIT 1, SPACE OVER EIGHT SPACES, AND THE TYPE THE DIGIT 1"
1710 PRINT "AGAIN."
1720 INPUT F$
1730 IF F$="1         1" THEN 1670
1740 NEXT I
1750 PRINT "                        FAILURE TO ENTER PROPER DATA."
1760 PRINT
1770 PRINT "                                    END TEST."
1780 PRINT
1790 PRINT
1800 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

```
PROGRAM FILE 83


                    SECTION 83.1

  EVALUATION OF A SUBSCRIPTED VARIABLE IN A VARIABLE LIST

                    BEGIN TEST.


     AFTER EACH SUCCESSIVE INPUT-PROMPT, ENTER SEQUENTIALLY
ONE OF THE FOLLOWING LISTED NUMBERS:1.5,2.5,3.5,4.5,5.5,
6.5,7.5,8.5,9.5,10.5
?1.5
?2.5
?3.5
?4.5
?5.5
?6.5
?7.5
?8.5
?9.5
?10.5
     LISTED BELOW ARE YOUR 10 INPUTTED DATA VALUES FOR THE
ARRAY A(I).  THE ELEMENTS ARE LISTED IN THE ORDER OF A(1),
A(2),A(3),...,A(10) FROM TOP TO BOTTOM RESPECTIVELY.

                                          1.5
                                          2.5
                                          3.5
                                          4.5
                                          5.5
                                          6.5
                                          7.5
                                          8.5
                                          9.5
                                          10.5

     AFTER THE INPUT-PROMPT, YOU ARE TO ENTER ONLY TWO IN-
PUT-VALUES.  (1) FOR THE FIRST INPUT-VALUE, SELECT ONE OF
THE DIGITS 1,2,3,...,10 AS THE SELECTION OF ONE OF THE ELE-
MENTS OF ARRAY A(I).  (2) FOR THE SECOND INPUT-VALUE, ENTER
THEN NUMBER 54.8.
?5,54.8
     LISTED BELOW ARE AGAIN THE ELEMENTS OF ARRAY A(I) IN
THE SAME ORDER A(1),A(2),A(3),...,A(10) FROM TOP TO BOTTOM
```

RESPECTIVELY, BUT IN THIS LIST THE VALUE OF YOUR SELECTED ELEMENT SHOULD BE 54.8 AND NOT THE VALUE IT HAD IN THE LIST ABOVE.

1.5
2.5
3.5
4.5
54.8
6.5
7.5
8.5
9.5
10.5

END TEST.

SECTION 83.2

ALLOWABLE CHARACTERS FOR UNQUOTED STRING INPUTS

BEGIN TEST.

AT EACH SUCCESSIVE INPUT-PROMPT, ENTER ONE OF THE ROWS OF CHARACTERS BELOW AS THEY APPEAR FROM TO TO BOTTOM:

ABCDEFGHIJKLM
NOPQRSTUVWXYZ
0123456789
*():$=><
- %;+.?/
?ABCDEFGHIJKLM
?NOPQRSTUVWXYZ
?0123456789
?*():$=><
?- %;+.?/

AFTER THE INPUT-PROMPT, ENTER THE DIGIT 1, SPACE OVER EIGHT SPACES, AND THEN TYPE THE DIGIT 1 AGAIN.
?1 1

UNQUOTED-STRING-CHARACTER= SPACE/PLAIN-STRING-CHARACTER

PLAIN-STRING-CHARACTERS.

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789*():$=><- %;+.?/

IF THE LINE OF CHARACTERS PRINTED BELOW, APPEAR IN THE SAME FORM AS THE LINE OF CHARACTERS ABOVE, CHECK TEST PASSED.

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789*():$=><- %;+.?/

SPACE (IMBEDDED).

IF AFTER THE COLUMN NUMBERED HEADINGS, THERE ARE EIGHT
COUNTABLE SPACES BETWEEN THE ONES- CHECK TEST PASSED.

```
000000000111111111122222222223333333333444444444455555555556666666666777
123456789012345678901234567890123456789012345678901234567890123456789012
1        1
```

END TEST.

84.0 INPUTTING MIXED DATA

84.1 Testing Leading and Trailing Spaces on Unquoted Strings

This test verifies that the implementation recognizes that leading and trailing spaces should be ignored when used with unquoted strings. The test has messages printed to the user. These instructions supply the information which the user should follow for each input-reply response. On output, the test generates two rows of digits to act as column labels. Following this output, the user inputs should be printed to verify that trailing and leading spaces are ignored by the implementation. The reader is referred to section 13.4 of BSR X3.60.

84.2 Inputting Leading and Trailing Spaces in Quoted Strings

This test shows that the implementation should recognize all leading and trailing spaces used in quoted strings. The test generates appropriate data input instructions to the user. These instructions supply the information the user should follow for each input-reply response. The test checks the correctness of the input values. Should the user fail to properly enter the data requested according to the printed instructions, the test will terminate its execution by printing the following message: FAILURE TO ENTER PROPER DATA. If the user correctly enters the data two rows of digits should be printed to act as column labels and the results entered by the user.

84.3 Inputting Mixed Data

This test shows that numeric constants and string constants can be entered in mixed form as the DATA-list for an input response. All data entered by the user will be checked for proper form as specified by the input-prompt messages. As in the previous tests, if any data is entered incorrectly, the user will be given three chances to enter the data correctly. If after the three chances the data is still entered incorrectly, the following message will be printed: FAILURE TO ENTER PROPER DATA and termination of the test will follow. However, if the data is entered correctly, an output should be printed in two columns. In the first column the required output will be printed and in the second column, the test system output will be printed.

```
**********************
* PROGRAM FILE 84 *
**********************
```

```
0010 PRINT "PROGRAM FILE 84"
0020 PRINT
```

```
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 84.1"
0100 PRINT
0110 PRINT "TESTING LEADING AND TRAILING SPACES ON UNQUOTED STRINGS."
0130 PRINT
0140 PRINT
0150 PRINT "                                BEGIN TEST."
0160 PRINT
0170 PRINT "NOTE:  INPUT SHOULD NOT EXCEED 72 CHARACTERS PER LINE."
0180 PRINT
0190 PRINT
0200 FOR I=1 TO 3
0210 PRINT "     AFTER THE INPUT-PROMPT, AS A CHOICE OF INPUT, MAKE"
0220 PRINT "YOUR SELECTION FROM THE DIGITS 0,1,2,3,...,9 IN ACCORDANCE"
0230 PRINT "WITH THE FOLLOWING STEPS."
0240 PRINT
0250 PRINT "     (1) SPACE OVER ABOUT FIVE SPACES, THEN TYPE YOUR"
0260 PRINT "SELECTED DIGIT."
0270 PRINT "     (2) SPACE OVER ABOUT FIVE SPACES, FOLLOW THIS ACTION"
0280 PRINT "BY TYPING THE COMMA PUNCTUATION MARK, THEN TYPE YOUR NEXT"
0290 PRINT "SELECTED DIGIT."
0300 INPUT G$,H$
0310 PRINT
0320 PRINT
0330 PRINT "     IF AFTER THE COLUMN NUMBERED HEADINGS, YOUR TWO"
0340 PRINT "SELECTED DIGITS ARE PRINTED STARTING IN COLUMN ONE WITHOUT"
0350 PRINT "ANY SPACES SEPARATING THEM, TYPE YES AFTER THE INPUT-PROMPT"
0360 PRINT "WHICH FOLLOWS, AND IF NOT, TYPE NO."
0370 PRINT
0380 PRINT
0390 PRINT "00000000011111111112222222223333333333444444444 5";
0400 PRINT "55555555566666666667777"
0410 PRINT "12345678901234567890123456789012345678901234567890";
0420 PRINT "123456789012345678901 2"
0430 PRINT G$;H$
0440 INPUT I$
0450 IF I$="YES" THEN 490
0460 GOSUB 540
0470 NEXT I
0480 GOTO 650
0490 PRINT "                          TEST PASSED."
0500 GOTO 520
0510 PRINT "                          TEST FAILED."
0520 PRINT
0530 GOTO 680
0540 PRINT
0550 PRINT "     IF THE OUTPUT WAS INCORRECT BECAUSE YOU DID NOT FOLLOW"
0560 PRINT "THE INPUT DIRECTIONS CORRECTLY, THEN YOU SHOULD TYPE YES"
0570 PRINT "WHEN THE INPUT-PROMPT APPEARS AFTER THIS PARAGRAPH AND BY"
0580 PRINT "DOING SO YOU WILL HAVE ANOTHER CHANCE TO ENTER DATA COR-"
0590 PRINT "RECTLY.  HOWEVER, IF YOU DID FOLLOW THE INPUT DIRECTIONS"
0600 PRINT "CORRECTLY, TYPE A NO WHEN THE INPUT-PROMPT APPEARS AFTER"
0610 PRINT "THIS PARAGRAPH."
0620 INPUT J$
0630 IF J$="NO" THEN 510
0640 RETURN
```

```
0650 PRINT
0660 PRINT "                    FAILURE TO ENTER DATA CORRECTLY."
0670 PRINT
0680 PRINT "                                END TEST."
0690 PRINT
0700 PRINT
0710 PRINT
0720 PRINT
0730 PRINT "                         SECTION 84.2"
0740 PRINT
0750 PRINT "INPUTTING LEADING AND TRAILING SPACES IN QUOTED STRINGS."
0760 PRINT
0770 PRINT
0780 PRINT "                                BEGIN TEST."
0790 PRINT
0800 PRINT "     AFTER THE INPUT-PROMPT, ENTER DATA WITHIN THE QUOTE-"
0810 PRINT "MARK SYMBOLS OF PUNCTUATION IN ACCORDANCE WITH THE FOLLOW-"
0820 PRINT "ING PROCEDURES:"
0830 PRINT
0840 PRINT "     (1) SPACE OVER EXACTLY FIVE SPACES, THEN TYPE THE WORD"
0850 PRINT "THAT FOLLOWS:  SPACED"
0860 PRINT "     (2) SPACE OVER EXACTLY FIVE SPACES, FOLLOW THIS ACTION"
0870 PRINT "BY TYPING THE COMMA PUNCTUATION MARK, THEN TYPE THE WORD"
0880 PRINT "THAT FOLLOWS:  APART"
0890 INPUT L$,M$
0900 GOSUB 1040
0910 PRINT
0920 PRINT "     IF AFTER THE COLUMN NUMBERED HEADINGS, THE WORD SPACED"
0930 PRINT "IS PRINTED STARTING IN THE SIXTH COLUMN AND THE WORD APART"
0940 PRINT "IS PRINTED STARTING IN THE SEVENTEENTH COLUMN, CHECK TEST"
0950 PRINT "PASSED."
0960 PRINT
0970 PRINT "00000000011111111112222222222333333333344444444445";
0980 PRINT "55555555566666666777"
0990 PRINT "12345678901234567890123456789012345678901234567890";
1000 PRINT "123456789012345678912"
1010 PRINT L$;M$
1020 PRINT
1030 GOTO 1220
1040 IF L$<>"     SPACED     "THEN 1070
1050 IF M$<>"APART" THEN 1070
1060 RETURN
1070 FOR I=1 TO 3
1080 PRINT "     DATA ERROR, PLEASE ENTER DATA AFTER THE INPUT-PROMPT"
1090 PRINT "WITHIN THE QUOTE-MARK SYMBOLS OF PUNCTUATION IN ACCORDANCE"
1100 PRINT "WITH THE FOLLOWING STEPS:"
1110 PRINT
1120 PRINT "     (1) SPACE OVER EXACTLY FIVE SPACES, THEN TYPE THE WORD"
1130 PRINT "THAT FOLLOWS:  SPACED"
1140 PRINT "     (2) SPACE OVER EXACTLY FIVE SPACES, FOLLOW THIS ACTION"
1150 PRINT "BY TYPING THE COMMA PUNCTUATION MARK, THEN TYPE THE WORD"
1160 PRINT "THAT FOLLOWS:  APART"
1170 INPUT L$,M$
1180 IF L$<>"     SPACED     "THEN 1200
1190 IF M$="APART" THEN 1060
1200 NEXT I
1210 PRINT "                    FAILURE TO ENTER PROPER DATA."
```

```
1220 PRINT
1230 PRINT "                                   END TEST."
1240 PRINT
1250 PRINT
1260 PRINT
1270 PRINT
1280 PRINT "                                SECTION 84.3"
1290 PRINT
1293 PRINT "                            INPUTTING MIXED DATA."
1295 PRINT
1300 PRINT "                                  BEGIN TEST."
1310 PRINT
1320 PRINT "     AFTER THE INPUT-PROPT, ENTER DATA ACCORDING TO THE"
1330 PRINT "STEPS WHICH FOLLOW, AND SEPARATING THE PERFORMANCE OF EACH"
1340 PRINT "STEP FROM EACH OTHER WITH THE COMMA PUNCTUATION MARK:"
1350 PRINT
1360 PRINT "     (1) WITHIN THE QUOTE-MARK SYMBOLS OF PUNCTUATION, TYPE"
1370 PRINT "THE WORD THAT FOLLOWS:  QUOTED"
1380 PRINT "     (2) TYPE THE NUMERAL THAT FOLLOWS:  1.23456"
1390 PRINT "     (3) TYPE THE WORD THAT FOLLOWS:  UNQUOTED"
1400 INPUT N$,O,P$
1410 GOSUB 1510
1420 PRINT
1430 PRINT "NEEDED","SYSTEM"
1440 PRINT "OUTPUT","OUTPUT"
1450 PRINT
1460 PRINT "QUOTED",N$
1470 PRINT " 1.23456 ",O
1480 PRINT "UNQUOTED",P$
1490 PRINT
1500 GOTO 1710
1510 IF N$<>"QUOTED" THEN 1550
1520 IF O<>1.23456 THEN 1550
1530 IF P$<>"UNQUOTED" THEN 1550
1540 RETURN
1550 FOR I=1 TO 3
1560 PRINT "     DATA ERROR, PLEASE ENTER DATA AFTER THE INPUT-PROMPT"
1570 PRINT "ACCORDING TO THE STEPS WHICH FOLLOW, AND SEPARATE THE PER-"
1580 PRINT "FORMANCE OF EACH STEP FROM EACH OTHER WITH THE COMMA PUNC-"
1590 PRINT "TUATION MARK:"
1600 PRINT
1610 PRINT "     (1) WITHIN THE QUOTE-MARK SYMBOLS OF PUNCTUATION, TYPE"
1620 PRINT "THEN WORD THAT FOLLOWS:  QUOTED"
1630 PRINT "     (2) TYPE THE NUMERAL THAT FOLLOWS:  1.23456"
1640 PRINT "     (3) TYPE THE WORD THAT FOLLOWS:  UNQUOTED"
1650 INPUT N$,O,P$
1660 IF N$<>"QUOTED" THEN 1690
1670 IF O<>1.23456 THEN 1690
1680 IF P$="UNQUOTED" THEN 1540
1690 NEXT I
1700 PRINT "                        FAILURE TO ENTER PROPER DATA."
1710 PRINT
1720 PRINT "                                  END TEST."
1730 PRINT
1740 PRINT
1750 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 84

SECTION 84.1

TESTING LEADING AND TRAILING SPACES ON UNQUOTED STRINGS.

BEGIN TEST.

NOTE: INPUT SHOULD NOT EXCEED 72 CHARACTERS PER LINE.

AFTER THE INPUT-PROMPT, AS A CHOICE OF INPUT, MAKE YOUR SELECTION FROM THE DIGITS 0,1,2,3,...,9 IN ACCORDANCE WITH THE FOLLOWING STEPS.

(1) SPACE OVER ABOUT FIVE SPACES, THEN TYPE YOUR SELECTED DIGIT.
(2) SPACE OVER ABOUT FIVE SPACES, FOLLOW THIS ACTION BY TYPING THE COMMA PUNCTUATION MARK, THEN TYPE YOUR NEXT SELECTED DIGIT.
? 2 ,8

IF AFTER THE COLUMN NUMBERED HEADINGS, YOUR TWO SELECTED DIGITS ARE PRINTED STARTING IN COLUMN ONE WITHOUT ANY SPACES SEPARATING THEM, TYPE YES AFTER THE INPUT-PROMPT WHICH FOLLOWS, AND IF NOT, TYPE NO.

```
000000000111111111122222222223333333333444444444455555555556666666667777
123456789012345678901234567890123456789012345678901234567890123456789012
28
?YES
```

TEST PASSED.

END TEST.

SECTION 84.2

INPUTTING LEADING AND TRAILING SPACES IN QUOTED STRINGS.

BEGIN TEST.

AFTER THE INPUT-PROMPT, ENTER DATA WITHIN THE QUOTE-MARK SYMBOLS OF PUNCTUATION IN ACCORDANCE WITH THE FOLLOWING PROCEDURES:

(1) SPACE OVER EXACTLY FIVE SPACES, THEN TYPE THE WORD THAT FOLLOWS: SPACED
(2) SPACE OVER EXACTLY FIVE SPACES, FOLLOW THIS ACTION BY TYPING THE COMMA PUNCTUATION MARK, THEN TYPE THE WORD THAT FOLLOWS: APART
?" SPACED ","APART"

IF AFTER THE COLUMN NUMBERED HEADINGS, THE WORD SPACED IS PRINTED STARTING IN THE SIXTH COLUMN AND THE WORD APART IS PRINTED STARTING IN THE SEVENTEENTH COLUMN, CHECK TEST PASSED.

```
000000000111111111122222222223333333333444444444455555555556666666666777
123456789012345678901234567890123456789012345678901234567890123456789012
     SPACED     APART
```

END TEST.

SECTION 84.3

INPUTTING MIXED DATA.

BEGIN TEST.

AFTER THE INPUT-PROPT, ENTER DATA ACCORDING TO THE STEPS WHICH FOLLOW, AND SEPARATING THE PERFORMANCE OF EACH STEP FROM EACH OTHER WITH THE COMMA PUNCTUATION MARK:

(1) WITHIN THE QUOTE-MARK SYMBOLS OF PUNCTUATION, TYPE THE WORD THAT FOLLOWS: QUOTED
(2) TYPE THE NUMERAL THAT FOLLOWS: 1.23456
(3) TYPE THE WORD THAT FOLLOWS: UNQUOTED
?"QUOTED",1.23456,UNQUOTED

```
NEEDED          SYSTEM
OUTPUT          OUTPUT

QUOTED          QUOTED
 1.23456         1.23456
UNQUOTED        UNQUOTED
```

END TEST.

85.0 EXCEPTION TEST - TYPE OF DATUM INCORRECT

This test verifies that the implementation recognizes when the type of a datum does not match the type of the variable to which it is assigned (see section 13.5 of BSR X3.60). If an improper datum type is entered, the implementation should consider this an exception. The recovery procedure requires that the user be allowed to resupply the input value.

The test begins with a message asking the user to enter, as an input-reply, the string of characters: SIX. This string is non-numeric, which is not the type the system expects. Thus, the implementation should request a resupply of data. If not, then the implementation does not meet the standard specifications. If the system does ask the user to resupply data, then the user should enter the numeric constant 1.

```
********************
* PROGRAM FILE 85 *
********************
```

```
0010 PRINT "PROGRAM FILE 85"
0020 PRINT
0030 PRINT
0040 PRINT
0300 PRINT "                          SECTION 85.0"
0310 PRINT
0320 PRINT "   TYPE OF DATUM INCORRECT FOR VARIABLE IT IS ASSIGNED TO."
0330 PRINT
0340 PRINT
0350 PRINT "*****NOTE:  FOR THIS PART OF THE TEST, TO STOP THE CONTINU-"
0360 PRINT "OUS REQUEST BY THE SYSTEM TO RESUPPLY THE INPUT-REPLY"
0370 PRINT "(IF IT SHOULD OCCUR), JUST ENTER UPON A REQUEST THE DIGIT 1."
380 PRINT "*****"
0390 PRINT
0400 PRINT
0410 PRINT "                                   BEGIN TEST."
0420 PRINT
0430 PRINT "AFTER THE INPUT-PROMPT, ENTER THE WORD: SIX"
0440 INPUT A
0450 IF A=1 THEN 480
0460 PRINT "TEST WAS NOT PROPERLY EXECUTED."
0470 GOTO 490
0480 PRINT "TEST WAS PROPERLY EXECUTED."
0490 PRINT
0500 PRINT "                                    END TEST."
0510 PRINT
0520 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

```
PROGRAM FILE 85


                        SECTION 85.0

   TYPE OF DATUM INCORRECT FOR VARIABLE IT IS ASSIGNED TO.


*****NOTE:  FOR THIS PART OF THE TEST, TO STOP THE CONTINU-
OUS REQUEST BY THE SYSTEM TO RESUPPLY THE INPUT-REPLY
(IF IT SHOULD OCCUR), JUST ENTER UPON A REQUEST THE DIGIT 1.
*****

                                BEGIN TEST.

AFTER THE INPUT-PROMPT, ENTER THE WORD: SIX
?SIX
?ILLEGAL DATA, PLEASE RESUPPLY
?1
TEST WAS PROPERLY EXECUTED.

                                END TEST.
```

86.0 EXCEPTION TEST FOR INPUT - TOO MUCH DATA IN DATA LIST

The objective of this test is to determine whether the implementation recognizes too much data in the input data-list as an exception (see section 13.5 of BSR X3.60). Upon recognition of the error, the implementation should allow the user to resupply his input data-list. The test has an instruction message that informs the user to input the data-list: 5, -35. The test, however, has only one numeric variable in its variable-list for the INPUT statement. Therefore the user should be requested, by the test system, to resupply the input-list. Once this message appears the test has been passed by the host system. In order to terminate the program the user should enter the constant 1.

```
********************
* PROGRAM FILE 86 *
********************
```

```
0010 PRINT "PROGRAM FILE 86"
0020 PRINT
0030 PRINT
0040 PRINT
0120 PRINT "                          SECTION 86.0"
0130 PRINT
0140 PRINT "                        TOO MUCH DATA IN DATA-LIST. "
0150 PRINT
0160 PRINT
0170 PRINT "*****NOTE:  FOR THIS PART OF THE TEST, TO STOP THE CONTINU-"
0180 PRINT "OUS REQUEST BY THE SYSTEM--THAT IS, IF IT SHOULD OCCUR--TO"
0190 PRINT "RESUPPLY THE INPUT-REPLY, JUST ENTER UPON A REQUEST THE"
0200 PRINT "DIGIT 1.*****"
0210 PRINT
0220 PRINT
0230 PRINT "                                    BEGIN TEST."
0240 PRINT
0250 PRINT "AFTER THE INPUT-PROMPT, ENTER THE NUMBERS AS FOLLOWS: 5,-35"
0260 LET A=9999
0270 INPUT A
0280 IF A=1 THEN 310
0290 PRINT "TEST WAS NOT EXECUTED PROPERLY."
0300 GOTO 320
0310 PRINT "TEST WAS EXECUTED PROPERLY."
0320 PRINT
0330 PRINT "                                     END TEST."
0340 PRINT
0350 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

```
PROGRAM FILE 86


                        SECTION 86.0

                  TOO MUCH DATA IN DATA-LIST.


*****NOTE:  FOR THIS PART OF THE TEST, TO STOP THE CONTINU-
OUS REQUEST BY THE SYSTEM--THAT IS, IF IT SHOULD OCCUR--TO
RESUPPLY THE INPUT-REPLY, JUST ENTER UPON A REQUEST THE
DIGIT 1.*****


                           BEGIN TEST.

AFTER THE INPUT-PROMPT, ENTER THE NUMBERS AS FOLLOWS: 5,-35
?5,-35
?TOO MUCH DATA. PLEASE RESUPPLY
?5,-35
?TOO MUCH DATA. PLEASE RESUPPLY
?1
TEST WAS EXECUTED PROPERLY

                           END TEST.
```

87.0 EXCEPTION TEST - INSUFFICIENT DATA IN DATA-LIST

The objective of this test is to determine whether the implementation will recognize too little data in the input-list as an exception (see section 13.5 of BSR X3.60). Upon recognition of the error, the implementation should allow the user to resupply the input data. The test has a message printed to the user to enter the number 64. However, the program requires more than one numeric variable in the input-list. Therefore there should be an implementation prompt to resupply the input-list. The user should continue to supply the value 64 several times to determine whether the system continues to prompt. Finally the user should enter the list: 1,2 in order to terminate the test.

```
********************
* PROGRAM FILE 87 *
********************
```

```
0010 PRINT "PROGRAM FILE 87"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 87.0"
0100 PRINT
0110 PRINT "                     INSUFFICIENT DATA IN DATA-LIST. "
0120 PRINT
0130 PRINT
0140 PRINT "*****NOTE:  FOR THIS PART OF THE TEST, TO STOP THE CONTINU-"
0150 PRINT "OUS REQUEST BY THE SYSTEM--THAT IS, IF IT SHOULD OCCUR--TO"
0160 PRINT "RESUPPLY THE INPUT-REPLY, JUST ENTER UPON A REQUEST THE"
0170 PRINT "DIGITS AS FOLLOWS: 1,2.*****"
0180 PRINT
0190 PRINT
0200 PRINT "                                  BEGIN TEST."
0210 PRINT
0220 PRINT "AFTER THE INPUT-PROMPT, ENTER THE FOLLOWING NUMBER: 64"
0230 INPUT A,B
0240 IF A<>1 THEN 260
0250 IF B=2 THEN 280
0260 PRINT "TEST WAS NOT EXECUTED PROPERLY."
0270 GOTO 290
0280 PRINT "TEST WAS EXECUTED PROPERLY."
0290 PRINT
0300 PRINT "                                    END TEST."
0310 PRINT
0320 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 87

SECTION 87.0

INSUFFICIENT DATA IN DATA-LIST.

*****NOTE: FOR THIS PART OF THE TEST, TO STOP THE CONTINUOUS REQUEST BY THE SYSTEM--THAT IS, IF IT SHOULD OCCUR--TO RESUPPLY THE INPUT-REPLY, JUST ENTER UPON A REQUEST THE DIGITS AS FOLLOWS: 1,2.*****

BEGIN TEST.

```
AFTER THE INPUT-PROMPT, ENTER THE FOLLOWING NUMBER: 64
?64
?INSUFFICIENT DATA IN LIST. PLEASE RESUPPLY.
?64
?INSUFFICIENT DATA IN LIST. PLEASE RESUPPLY.
?64
?INSUFFICIENT DATA IN LIST. PLEASE RESUPPLY.
?1,2
TEST WAS EXECUTED PROPERLY.
```

END TEST.

88.0 NUMERIC UNDERFLOW ON INPUT

The objective of this test is to determine whether the implementation will diagnose as an exception the input of a numeric value too small to be represented by the test system (see section 13.4 of BSR X3.60). Upon recognition of the exception,the implementation should replace the value with zero and continue the program. The test has a message printed asking the user to input the numerical value of 10.0E-99999. After entering this value the system will test the input variable to determine whether it has been assigned the value 0. If not, then a message will indicate that the program did not execute properly.

****
* PROGRAM FILE 88 *
****

```
0010 PRINT "PROGRAM FILE 88"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 88.0"
0100 PRINT
0110 PRINT "   THE CONVERSION OF A NUMERIC DATUM CAUSES AN UNDERFLOW. "
0120 PRINT
0130 PRINT
0190 PRINT
0200 PRINT "                                  BEGIN TEST."
0210 PRINT
0220 PRINT "AFTER THE APPEARANCE OF THE INPUT-PROMPT, ENTER THE FOLLOW-"
0230 PRINT "ING NUMBER: 10.0E-99999"
0240 INPUT A
0250 IF A=0 THEN 280
0260 PRINT "TEST WAS NOT EXECUTED PROPERLY."
0270 GOTO 290
0280 PRINT "TEST WAS EXECUTED PROPERLY."
0290 PRINT
0300 PRINT "                                   END TEST."
0310 PRINT
0320 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 88

SECTION 88.0

THE CONVERSION OF A NUMERIC DATUM CAUSES AN UNDERFLOW.

BEGIN TEST.

AFTER THE APPEARANCE OF THE INPUT-PROMPT, ENTER THE FOLLOW-
ING NUMBER: 10.0E-99999
?10.0E-99999
TEST WAS EXECUTED PROPERLY.

END TEST.

89.0 EXCEPTION TEST - NUMERIC OVERFLOW

The objective of this test is to determine whether the implementation will recognize, as an exception, a numeric value in an input-list that is too large to be represented in the test system (see section 13.5 of BSR X3.60). Upon recognition of the error, the implementation should allow the user to resupply the input-list. The test has an instruction printed to the user to enter the value 9.99999E99999. As in section 88.0, the system should prompt the user to resupply data since an overflow was registered. Again in order to terminate the system request the user should type 1.

****
* PROGRAM FILE 89 *
****

```
0010 PRINT "PROGRAM FILE 89"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 89.0"
0100 PRINT
0110 PRINT "   THE CONVERSION OF A NUMERIC DATUM CAUSES AN OVERFLOW. "
0120 PRINT
0130 PRINT
0140 PRINT "*****NOTE:  FOR THIS PART OF THE TEST, TO STOP THE CONTINU-"
0150 PRINT "OUS REQUEST BY THE SYSTEM--THAT IS IF IT SHOULD OCCUR--TO"
0160 PRINT "RESUPPLY THE INPUT-RELY, JUST ENTER UPON A RE )UEST THE"
0170 PRINT "DIGIT 1.*****"
0180 PRINT
0190 PRINT
0200 PRINT "                                  BEGIN TEST."
0210 PRINT
0220 PRINT "AFTER THE APPEARANCE OF THE INPUT-PROMPT, ENTER THE FOLLOW-"
0230 PRINT "ING NUMBER: 9.99999E99999"
0240 INPUT A
0250 IF A=1 THEN 280
0260 PRINT "TEST WAS NOT EXECUTED PROPERLY."
0270 GOTO 290
0280 PRINT "TEST WAS EXECUTED PROPERLY."
0290 PRINT
0300 PRINT "                                    END TEST."
0310 PRINT
0320 END
```

****

```
                    * SAMPLE OUTPUT *
                    ******************



PROGRAM FILE 89


                         SECTION 89.0

   THE CONVERSION OF A NUMERIC DATUM CAUSES AN OVERFLOW.


*****NOTE:  FOR THIS PART OF THE TEST, TO STOP THE CONTINU-
OUS REQUEST BY THE SYSTEM--THAT IS IF IT SHOULD OCCUR--TO
RESUPPLY THE INPUT-RELY, JUST ENTER UPON A REQUEST THE
DIGIT 1.*****


                         BEGIN TEST.

AFTER THE APPEARANCE OF THE INPUT-PROMPT, ENTER THE FOLLOW-
ING NUMBER: 9.99999E99999
?9.99999E99999
?OVERFLOW IN INPUT DATA. PLEASE RETYPE
?9.99999E99999
?OVERFLOW IN INPUT DATA. PLEASE RETYPE
?1.0
TEST WAS EXECUTED PROPERLY.

                         END TEST.
```

90.0 TESTING THE INT AND SGN FUNCTIONS

90.1 The INT Function

The objective of this test is to use the INT function which returns the largest integer not greater than the argument specified (see section 8 of BSR X3.60). The test uses simple numeric-variables as the argument for the function. Assignments of various numerical values are made to the simple variable used as the argument parameter. On output, the test has three column labels printed. In sequential order the labels should read as follows: ARGUMENT, NEEDED EVALUATION, ACTUAL EVALUATION. The first column lists the values assigned to the function argument. The second lists the required evaluations by INT for each argument. Finally, the third column lists those values generated by the implementation-supplied INT function.

90.2 The SGN Function

This test initiates the use of the SGN function, which supplies -1 if the argument is negative, 1 if the argument is positive and 0 if the argument is 0. This test, except for the use of the SGN function instead of the INT function, is similar to section 90.1 and has a similarly formatted output.

```
********************
* PROGRAM FILE 90 *
********************
```

```
0010 PRINT "PROGRAM FILE 90"
0020 PRINT
0030 PRINT
0040 PRINT
0050 PRINT "                              SECTION 90.0."
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT
0100 PRINT "    (TESTING APPLICATIONSS OF THE INT AND SGN FUNCTION.)"
0110 PRINT
0120 PRINT
0130 PRINT
0140 PRINT "                     SECTION 90.1: THE INT FUNCTION."
0150 PRINT
0160 PRINT "                                BEGIN TEST."
0170 PRINT
0180 PRINT
0190 PRINT "     EACH EVALUATED FAILURE OF THE INT FUNCTION WILL BE DE-"
0200 PRINT "NOTED BY AN ASTERISK BEING PRINTED ON THAT COMPARATIVE ROW"
```

```
0210 PRINT "OF OUTPUT.  CHECK TEST PASSED IF THERE ARE NOT ANY ASTER-"
0220 PRINT "ISKS."
0230 PRINT
0240 PRINT
0250 PRINT " ","  NEEDED  ","  ACTUAL  "
0260 PRINT " ARGUMENT ","EVALUATION","EVALUATION"
0270 PRINT
0280 FOR I=1 TO 10
0290 READ M,N
0300 LET X=INT(N)
0310 IF X<>M THEN 340
0320 PRINT N,M,X
0330 GOTO 350
0340 PRINT N,M,X;"*"
0350 NEXT I
0360 PRINT
0370 DATA 1,1.99999,0,.987654,12345,12345.6
0380 DATA -1,-.339872,-8,-7.2,-98766,-98765.4
0390 DATA -10,-9.99999,127,127.999,-128,-127.999
0400 DATA -1,-.987654
0410 PRINT
0420 PRINT
0430 PRINT
0440 PRINT "                    SECTION 90.2: THEN SGN FUNCTION."
0450 PRINT
0460 PRINT "                                BEGIN TEST."
0470 PRINT
0480 PRINT
0490 PRINT "     EACH EVALUATED FAILURE OF THE SGN FUNCTION WILL BE DE-"
0500 PRINT "NOTED BY AN ASTERISK BEING PRINTED ON THAT COMPARATIVE ROW"
0510 PRINT "OF OUTPUT.  CHECK TEST PASSED IF THERE ARE NOT ANY ASTER-"
0520 PRINT "ISKS."
0530 PRINT
0540 PRINT
0550 PRINT " ","  NEEDED  ","  ACTUAL  "
0560 PRINT " ARGUMENT ","EVALUATION","EVALUATION"
0570 PRINT
0580 FORI=1 TO 10
0590 READ M1,N1
0600 LET X1=SGN(N1)
0610 IF X1<>M1 THEN 640
0620 PRINT N1,M1,X1
0630 GOTO 650
0640 PRINT N1,M1,X1;"*"
0650 NEXT I
0660 PRINT
0670 DATA -1,-.930896,1,12345,-1,-1.222
0680 DATA 0,0.00000,1,2.18765,-1,-0.00023
0690 DATA 1,99999,0,0,-1,-888.99
0700 DATA 1,1.99999
0710 PRINT
0720 PRINT
0730 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 90

SECTION 90.0.

(TESTING APPLICATIONS OF THE INT AND SGN FUNCTIONS.)

SECTION 90.1: THE INT FUNCTION.

BEGIN TEST.

EACH EVALUATED FAILURE OF THE INT FUNCTION WILL BE DENOTED BY AN ASTERISK BEING PRINTED ON THAT COMPARATIVE ROW OF OUTPUT. CHECK TEST PASSED IF THERE ARE NOT ANY ASTERISKS.

ARGUMENT	NEEDED EVALUATION	ACTUAL EVALUATION
1.99999	1	1
0.987654	0	0
12345.6	12345	12345
-0.339872	-1	-1
-7.2	-8	-8
-98765.4	-98766	-98766
-9.99999	-10	-10
127.999	127	127
-127.999	-128	-128
-0.987654	-1	-1

SECTION 90.2: THEN SGN FUNCTION.

BEGIN TEST.

EACH EVALUATED FAILURE OF THE SGN FUNCTION WILL BE DENOTED BY AN ASTERISK BEING PRINTED ON THAT COMPARATIVE ROW OF OUTPUT. CHECK TEST PASSED IF THERE ARE NOT ANY ASTER-

ISKS.

ARGUMENT	NEEDED EVALUATION	ACTUAL EVALUATION
-0.930896	-1	-1
12345	1	1
-1.222	-1	-1
0	0	0
2.18765	1	1
-2.30000E-4	-1	-1
99999	1	1
0	0	0
-888.99	-1	-1
1.99999	1	1

91.0 PRINTING STRINGS BEYOND THE MARGIN

The objective of this section is to determine how the implementation handles printing beyond its specified margin. Since the margin is implementation defined this section uses specifically constructed long strings, tabbed over a large number of spaces in some of the tests, in order to take into account variable margin specifications. The reader is referred to section 12.4 of BSR X3.60.

91.1 Printing Concatenated Strings

The objective of this test section is to determine how the print delimiters effect printing strings of characters beyond the implementation-defined margin for a given system.

91.1.1 Using Semicolons With Quoted Strings

The objective of this test is to verify that the implementation, upon the evaluation of any print item, which generates a string whose length is greater than the implementation-defined margin, will generate an end-of-line each time the columnar position of the current line exceeds the margin. Printing then must begin at the first position in the next line. The test has several print statements that output string constants of 50 characters each. Each print-list ends with a semicolon, except for the last. The exact output for this test will depend on the implementation-defined margin; however, the output should show a continuous string of digits up to and including the last columnar position for the implementation-defined margin. If there are any characters remaining they must begin in column one of the next line.

91.1.2 Using Commas With Quoted Strings

The objective of this test is similar to section 91.1.1. In this test, the print delimiter used is a comma. However, there is a significant amount of difference between the two outputs. For this test the string of digits should not be continuous (i.e., there should be spaces within the string of printed digits). The number of spaces depends on the implementation-defined zone lengths. The output must show that the digits should be printed up to and including the last columnar position for a defined margin, with printing then continuing on the next line.

91.1.3 Using Semicolons With Assigned Strings

The objective of this test is to determine whether the assignment of strings that exceed the margin will affect the evaluation of a print-item. The print-list in this exercise generates a string, whose length is greater than the defined margin. The implementation should still generate an end-of-line each time the columnar position of the current line exceeds the margin. The test begins by assigning strings of various lengths to string variables. These strings vary from lengths of 1 character to 18 characters. The assigned variables are then printed using semicolon delimiters. The output for this test should generate a similar display to that described in

section 91.1.1.

91.1.4 Using Commas With Assigned Strings

The objective of this test is similar to section 91.1.3, except, in this test the print separator used in the variable-list for the PRINT statement is the comma. The structure of this test, except for the comma as print-separator, is similar to the structure of test section 91.1.3, however, the strings assigned to the string variables are all of equal length. The output for this test should appear similar to that of section 91.1.2.

91.2 Simple TAB Tests Beyond The Margin

The objective of this test section is to determine whether the implementation recognizes the standard specified relationship between the TAB argument and the implementation-defined margin .

91.2.1 TAB Argument is Less Than the Value of the Current Print Position

The objective of this test is to determine whether the implementation will, upon evaluating a TAB argument whose rounded value is less than the columnar position of the current line, generate an end-of-line and enough spaces to set the columnar position of the new current line to the required position. This test prints a message that there should be no printed characters after the period at the end of the message. If there are, then the tabbing has failed this test.

91.2.2 TAB Assigned Strings Less Than Current Position

The objective of this test, except for assigning the string of characters to be tabbed to a string variable, is the same as that for section 91.2.1. In terms of output there should be no difference between this section and the output of section 91.2.1.

```
********************
* PROGRAM FILE 91 *
********************
```

```
0010 PRINT "PROGRAM FILE 91"
0020 PRINT
0030 PRINT
0040 PRINT
0120 PRINT "                  SECTION 91.1: CONCATENATED STRINGS."
0130 PRINT
0140 PRINT
0150 PRINT "           SECTION 91.1.1: USING SEMICOLON, QUOTED STRINGS."
```

```
0160 PRINT
0170 PRINT
0180 PRINT
0190 PRINT "                                    BEGIN TEST."
0200 PRINT
0210 PRINT "1234567890123456789012345678901234567890123456789012345678901234567890";
0220 PRINT "1234567890123456789012345678901234567890123456789012345678901234567890";
0230 PRINT "1234567890123456789012345678901234567890123456789012345678901234567890";
0240 PRINT "1234567890123456789012345678901234567890123456789012345678901234567890";
0250 PRINT "1234567890123456789012345678901234567890123456789012345678901234567890"
0260 PRINT
0270 PRINT "                                     END TEST."
0280 PRINT
0290 PRINT
0300 PRINT
0310 PRINT "             SECTION 91.1.2: USING COMMA, QUOTED STRINGS."
0320 PRINT
0330 PRINT
0340 PRINT
0350 PRINT "                                    BEGIN TEST."
0360 PRINT
0370 PRINT "1234567890123456789012345678901234567890123456789012345678901234567890",
0380 PRINT "1234567890123456789012345678901234567890123456789012345678901234567890",
0390 PRINT "1234567890123456789012345678901234567890123456789012345678901234567890",
0400 PRINT "1234567890123456789012345678901234567890123456789012345678901234567890",
0410 PRINT "1234567890123456789012345678901234567890123456789012345678901234567890"
0420 PRINT
0430 PRINT "                                     END TEST."
0440 PRINT
0450 PRINT
0460 PRINT
0470 PRINT "          SECTION 91.1.3: USING SEMICOLON, ASSIGNED STRINGS."
0480 PRINT
0490 PRINT
0500 PRINT
0510 PRINT "                                    BEGIN TEST."
0520 PRINT
0530 LET A$=" "
0540 LET B$="1"
0550 LET C$="12"
0560 LET D$="123"
0570 LET E$="1234"
0580 LET F$="12345"
0590 LET G$="123456"
0600 LET H$="1234567"
0610 LET I$="12345678"
0620 LET J$="123456789"
0630 LET K$="1234567890"
0640 LET L$="12345678901"
0650 LET M$="123456789012"
0660 LET N$="1234567890123"
0670 LET O$="12345678901234"
0680 LET P$="123456789012345"
0690 LET Q$="1234567890123456"
0700 LET R$="12345678901234567"
0710 LET S$="123456789012345678"
0720 PRINT
```

```
0730 PRINT A$;B$;C$;D$;E$;F$;G$;H$;I$;J$;K$;L$;M$;N$;O$;P$;Q$;R$;S$
0740 PRINT
0750 PRINT "                                   END TEST."
0760 PRINT
0770 PRINT
0780 PRINT
0790 PRINT "            SECTION 91.1.4: USING COMMA, ASSIGNED STRINGS."
0800 PRINT
0810 PRINT
0820 PRINT
0830 PRINT "                                 BEGIN TEST."
0840 PRINT
0850 LET A$="1234567890"
0860 LET B$="1234567890"
0870 LET C$="1234567890"
0880 LET D$="1234567890"
0890 LET E$="1234567890"
0900 LET F$="1234567890"
0910 LET G$="1234567890"
0920 LET H$="1234567890"
0930 LET I$="1234567890"
0940 LET J$="1234567890"
0950 LET K$="1234567890"
0960 LET L$="1234567890"
0970 LET M$="1234567890"
0980 LET N$="1234567890"
0990 PRINT "00000000011111111112222222222333333333344444444445";
1000 PRINT "55555555566666666677"
1010 PRINT "12345678901234567890123456789012345678901234567890";
1020 PRINT "123456789012345678912"
1030 PRINT A$,B$,C$,D$,E$,F$,G$,H$,I$,J$,K$,L$,M$,N$
1040 PRINT
1050 PRINT "                                   END TEST."
1060 PRINT
1070 PRINT
1080 PRINT
1090 PRINT "           SECTION 91.2: TAB-CALL WITHIN AND BEYOND MARGIN."
1100 PRINT
1110 PRINT
1120 PRINT "    SECTION 91.2.1: TABBING QUOTED STRINGS WHEN TAB-CALL IS"
1130 PRINT "                      LESS-THAN THE CURRENT PRINT POSITION."
1140 PRINT
1150 PRINT
1160 PRINT
1170 PRINT "                                 BEGIN TEST."
1180 PRINT
1190 PRINT "00000000011111111112222222222333333333344444444445";
1200 PRINT "55555555566666666677"
1210 PRINT "12345678901234567890123456789012345678901234567890";
1220 PRINT "123456789012345678912"
1230 PRINT "NO PRINT, THIS LINE, AFTER PERIOD.";TAB(15);"X"
1240 PRINT
1250 PRINT "                  X SHOULD BE ON LINE FOUR, COLUMN 15."
1260 PRINT
1270 PRINT "                                   END TEST."
1280 PRINT
1290 PRINT
```

```
1300 PRINT
1310 PRINT "  SECTION 91.2.2: TABBING ASSIGNED STRINGS WHEN TAB-CALL IS"
1320 PRINT "                  LESS-THAN THE CURRENT PRINT POSITION."
1330 PRINT
1340 PRINT
1350 PRINT
1360 PRINT "                                  BEGIN TEST."
1370 PRINT
1380 LET A$="X"
1390 PRINT "000000000011111111112222222222333333333344444444445";
1400 PRINT "55555555566666666677"
1410 PRINT "12345678901234567890123456789012345678901234567890";
1420 PRINT "1234567890123456789012"
1430 PRINT "NO PRINT, THIS LINE, AFTER PERIOD.";TAB(15);A$
1440 PRINT
1450 PRINT "                  X SHOULD BE ON LINE FOUR, COLUMN 15."
1460 PRINT
1470 PRINT "                                   END TEST."
1480 PRINT
1490 PRINT
1500 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

```
PROGRAM FILE 91

                SECTION 91.1: CONCATENATED STRINGS.

           SECTION 91.1.1: USING SEMICOLON, QUOTED STRINGS.

                              BEGIN TEST.

12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

                               END TEST.

           SECTION 91.1.2: USING COMMA, QUOTED STRINGS.
```

BEGIN TEST.

```
12345678901234567890123456789012345678901234567890      12345678901234
56789012345678901234567890123456789012345678901234567890        123456
78901234567890123456789012345678901234567890        1234567890123456
789012345678901234567890123456789012345678901234567890
```

END TEST.

SECTION 91.1.3: USING SEMICOLON, ASSIGNED STRINGS.

BEGIN TEST.

```
 1121231234123451234561234567123456781234567891234567890123456789011234
56789012123456789012312345678901234123456789012345123456789012345612345
67890123456712345678901234567
```

END TEST.

SECTION 91.1.4: USING COMMA, ASSIGNED STRINGS.

BEGIN TEST.

```
00000000011111111112222222222333333333344444444445555555555666666666677
12345678901234567890123456789012345678901234567890123456789012345678901
1234567890    1234567890    1234567890    1234567890    1234567890
1234567890    1234567890    1234567890    1234567890    1234567890
1234567890    1234567890    1234567890    1234567890
```

END TEST.

SECTION 91.2: TAB-CALL WITHIN AND BEYOND MARGIN.

SECTION 91.2.1: TABBING QUOTED STRINGS WHEN TAB-CALL IS LESS-THAN THE CURRENT PRINT POSITION.

BEGIN TEST.

```
00000000011111111112222222222333333333344444444445555555555666666666677
12345678901234567890123456789012345678901234567890123456789012345678901
NO PRINT, THIS LINE, AFTER PERIOD.
             X
```

X SHOULD BE ON LINE FOUR, COLUMN 15.

END TEST.

SECTION 91.2.2: TABBING ASSIGNED STRINGS WHEN TAB-CALL IS
LESS-THAN THE CURRENT PRINT POSITION.

BEGIN TEST.

```
000000000011111111112222222222333333333344444444445555555555666666666677
123456789012345678901234567890123456789012345678901234567890123456789012
NO PRINT, THIS LINE, AFTER PERIOD.
              X
```

X SHOULD BE ON LINE FOUR, COLUMN 15.

END TEST.

92.0 TABBING STRINGS BEYOND THE MARGIN

92.1 Quoted Strings

The objective of this test is to determine whether the implementation will, upon evaluating a TAB argument whose rounded value is greater than the implementation defined margin, reduce the value of n by an integral multiple of m so that it is in the range 1<=n<=m. In particular, n should be set equal to n-m*INT((n-1)/m). The reader is referred to section 12.4 of BSR X3.60. The test begins by requesting that the user input the value for the implementation-defined margin. The test then sequentially tabs 100 simple one character strings. Each TAB argument is increased by a value of 13 (i.e., the arguments are sequentially ordered 13, 26, 39, ..., 1300). On output, the test generates column count indices in order to verify each columnar position. Each printout should be sequentially labelled by the digits 1, 2, 3, ..., 100. Upon completion of the tab-call printout, messages should be printed out indicating in which columns the strings should begin.

92.2 Assigned Strings

The objective of this test, except for the assigning of the string of characters, that are to be tabbed, to a string variable, is the same as the objective stated for test section 92.1. This test uses READ/DATA statements to assign the strings A1, A2, A3, ..., A100 to the string variable A$ which is the print-item used in conjunction with the tab-calls. As in test section 92.1, 100 tabbings are used with the same value increment for the tab arguments. On output this test has a similar format to test 92.1.

```
********************
* PROGRAM FILE 92 *
********************
```

```
0010 PRINT "PROGRAM FILE 92"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT " SECTION 92.1: TABBING QUOTED STRINGS BEYOND THE CURRENT"
0100 PRINT "                 PRINT POSITION AS WELL AS BEYOND THE MAR-"
0110 PRINT "                 GIN."
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "                                BEGIN TEST."
0160 PRINT
0170 PRINT "     NOTE:  IN ORDER TO GET A PROPER OUTPUT, PLEASE ENTER"
0180 PRINT "THE MARGIN VALUE FOR THIS SYSTEM AFTER THE INPUT-PROMPT"
0200 INPUT M
```

```
0210 DIM A(100)
0220 GOSUB 720
0230 LET N=0
0240 FOR I=1 TO 100
0250 LET N=N+13
0260 PRINT TAB(N);"X";I
0270 LET A(I)=N-M*INT((N-1)/M)
0280 NEXT I
0290 GOSUB 720
0300 PRINT
0310 FOR I=1 TO 100
0320 PRINT "X";I;"SHOULD APPEAR BEGINNING IN COLUMN";A(I);"."
0330 NEXT I
0340 PRINT
0350 PRINT "                                    END TEST."
0360 PRINT
0370 PRINT
0380 PRINT
0390 PRINT "SECTION 92.2: TABBING ASSIGNED STRINGS BEYOND THE CURRENT"
0400 PRINT "                PRINT POSITION AS WELL AS BEYOND THE MAR-"
0410 PRINT "                GIN."
0420 PRINT
0430 PRINT
0440 PRINT
0450 PRINT "                                  BEGIN TEST."
0460 PRINT
0470 GOSUB 720
0480 LET N=0
0490 DATA A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16
0500 DATA A17,A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30
0510 DATA A31,A32,A33,A34,A35,A36,A37,A38,A39,A40,A41,A42,A43,A44
0520 DATA A45,A46,A47,A48,A49,A50,A51,A52,A53,A54,A55,A56,A57,A58
0530 DATA A59,A60,A61,A62,A63,A64,A65,A66,A67,A68,A69,A70,A71,A72
0540 DATA A73,A74,A75,A76,A77,A78,A79,A80,A81,A82,A83,A84,A85,A86
0550 DATA A87,A88,A89,A90,A91,A92,A93,A94,A95,A96,A97,A98,A99,A100
0560 FOR I=1 TO 100
0570 READ A$
0580 LET N=N+13
0590 PRINT TAB(N);A$
0600 LET A(I)=N-M*INT((N-1)/M)
0610 NEXT I
0620 GOSUB 720
0630 PRINT
0640 RESTORE
0650 FOR I=1 TO 100
0660 READ A$
0670 PRINT A$;" SHOULD BEGIN IN COLUMN";A(I);"."
0680 NEXT I
0690 PRINT
0700 PRINT "                                    END TEST."
0710 GOTO 1470
0720 FOR I=1 TO M
0730 ON 1+INT(I/100) GOTO 740,760,780,800,820,840,860,880,900,920
0740 LET A$="0"
0750 GOTO 930
0760 LET A$="1"
0770 GOTO 930
```

```
0780 LET A$="2"
0790 GOTO 930
0800 LET A$="3"
0810 GOTO 930
0820 LET A$="4"
0830 GOTO 930
0840 LET A$="5"
0850 GOTO 930
0860 LET A$="6"
0870 GOTO 930
0880 LET A$="7"
0890 GOTO 930
0900 LET A$="8"
0910 GOTO 930
0920 LET A$="9"
0930 PRINT A$;
0940 NEXT I
0950 FOR I=1 TO M
0960 LET P=(1+INT(I/10)-10*(INT(I/100)))
0970 ON P GOTO 980,1000,1020,1040,1060,1080,1100,1120,1140,1160
0980 LET A$="0"
0990 GOTO 1170
1000 LET A$="1"
1010 GOTO 1170
1020 LET A$="2"
1030 GOTO 1170
1040 LET A$="3"
1050 GOTO 1170
1060 LET A$="4"
1070 GOTO 1170
1080 LET A$="5"
1090 GOTO 1170
1100 LET A$="6"
1110 GOTO 1170
1120 LET A$="7"
1130 GOTO 1170
1140 LET A$="8"
1150 GOTO 1170
1160 LET A$="9"
1170 PRINT A$;
1180 NEXT I
1190 LET N=0
1200 FOR I=1 TO M
1210 LET J=I-1
1220 LET N=N+1
1230 ON N-10*(INT(J/10)) GOTO 1240,1260,1280,1300,1320,1340,1360,1380,1400
1240 LET A$="1"
1250 GOTO 1430
1260 LET A$="2"
1270 GOTO 1430
1280 LET A$="3"
1290 GOTO 1430
1300 LET A$="4"
1310 GOTO 1430
1320 LET A$="5"
1330 GOTO 1430
1340 LET A$="6"
```

```
1350 GOTO 1430
1360 LET A$="7"
1370 GOTO 1430
1380 LET A$="8"
1390 GOTO 1430
1400 LET A$="9"
1410 GOTO 1430
1420 LET A$="0"
1430 PRINT A$;
1440 NEXT I
1450 PRINT
1460 RETURN
1470 PRINT
1480 END
```

****
* SAMPLE OUTPUT *
****

PROGRAM FILE 92

SECTION 92.1: TABBING QUOTED STRINGS BEYOND THE CURRENT PRINT POSITION AS WELL AS BEYOND THE MARGIN.

BEGIN TEST.

NOTE: IN ORDER TO GET A PROPER OUTPUT, PLEASE ENTER THE MARGIN VALUE FOR THIS SYSTEM AFTER THE INPUT-PROMPT
?72

```
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000001111111111222222222233333333334444444444555555555566666666667777
1234567890123456789012345678901234567890123456789012345678901234567890123456789012
```

X 1

X 2

X 3

X 4

X 5

X 6

X 7

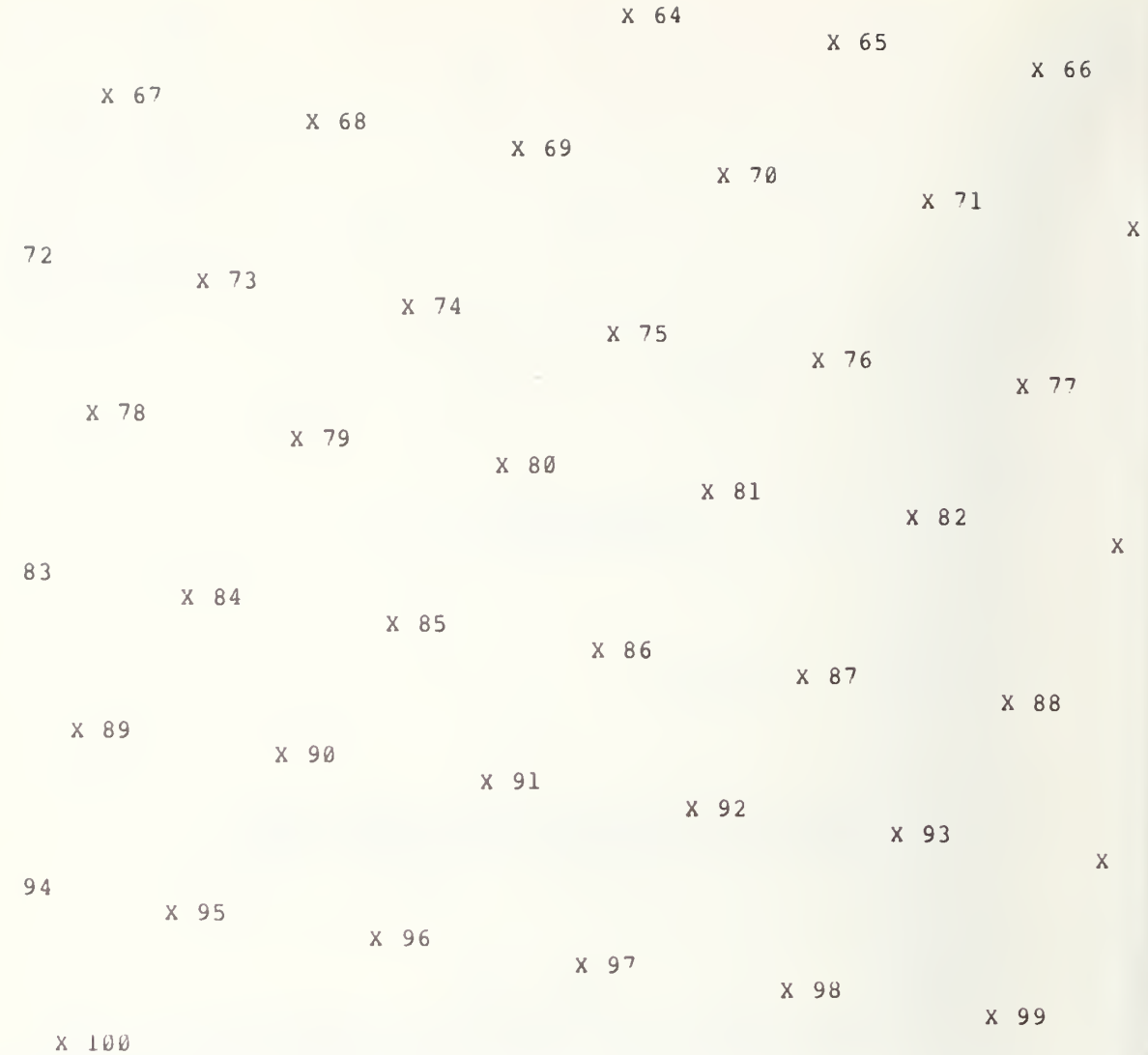
X 8

X 9

X 10

X


11
X 12
X 13
X 14
X 15
X 16
X 17
X 18
X 19
X 20
X 21
X
22
X 23
X 24
X 25
X 26
X 27
X 28
X 29
X 30
X 31
X 32
X
33
X 34
X 35
X 36
X 37
X 38
X 39
X 40
X 41
X 42
X 43
X
44
X 45
X 46
X 47
X 48
X 49
X 50
X 51
X 52
X 53
X 54
X
55
X 56
X 57
X 58
X 59
X 60
X 61
X 62
X 63




```
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000001111111111222222222233333333334444444444555555555566666666667777
1234567890123456789012345678901234567890123456789012345678901234567890123
```

X 1 SHOULD APPEAR BEGINNING IN COLUMN 13 .
X 2 SHOULD APPEAR BEGINNING IN COLUMN 26 .
X 3 SHOULD APPEAR BEGINNING IN COLUMN 39 .
X 4 SHOULD APPEAR BEGINNING IN COLUMN 52 .
X 5 SHOULD APPEAR BEGINNING IN COLUMN 65 .
X 6 SHOULD APPEAR BEGINNING IN COLUMN 6 .
X 7 SHOULD APPEAR BEGINNING IN COLUMN 19 .
X 8 SHOULD APPEAR BEGINNING IN COLUMN 32 .
X 9 SHOULD APPEAR BEGINNING IN COLUMN 45 .
X 10 SHOULD APPEAR BEGINNING IN COLUMN 58 .
X 11 SHOULD APPEAR BEGINNING IN COLUMN 71 .
X 12 SHOULD APPEAR BEGINNING IN COLUMN 12 .

X 13 SHOULD APPEAR BEGINNING IN COLUMN 25 .
X 14 SHOULD APPEAR BEGINNING IN COLUMN 38 .
X 15 SHOULD APPEAR BEGINNING IN COLUMN 51 .
X 16 SHOULD APPEAR BEGINNING IN COLUMN 64 .
X 17 SHOULD APPEAR BEGINNING IN COLUMN 5 .
X 18 SHOULD APPEAR BEGINNING IN COLUMN 18 .
X 19 SHOULD APPEAR BEGINNING IN COLUMN 31 .
X 20 SHOULD APPEAR BEGINNING IN COLUMN 44 .
X 21 SHOULD APPEAR BEGINNING IN COLUMN 57 .
X 22 SHOULD APPEAR BEGINNING IN COLUMN 70 .
X 23 SHOULD APPEAR BEGINNING IN COLUMN 11 .
X 24 SHOULD APPEAR BEGINNING IN COLUMN 24 .
X 25 SHOULD APPEAR BEGINNING IN COLUMN 37 .
X 26 SHOULD APPEAR BEGINNING IN COLUMN 50 .
X 27 SHOULD APPEAR BEGINNING IN COLUMN 63 .
X 28 SHOULD APPEAR BEGINNING IN COLUMN 4 .
X 29 SHOULD APPEAR BEGINNING IN COLUMN 17 .
X 30 SHOULD APPEAR BEGINNING IN COLUMN 30 .
X 31 SHOULD APPEAR BEGINNING IN COLUMN 43 .
X 32 SHOULD APPEAR BEGINNING IN COLUMN 56 .
X 33 SHOULD APPEAR BEGINNING IN COLUMN 69 .
X 34 SHOULD APPEAR BEGINNING IN COLUMN 10 .
X 35 SHOULD APPEAR BEGINNING IN COLUMN 23 .
X 36 SHOULD APPEAR BEGINNING IN COLUMN 36 .
X 37 SHOULD APPEAR BEGINNING IN COLUMN 49 .
X 38 SHOULD APPEAR BEGINNING IN COLUMN 62 .
X 39 SHOULD APPEAR BEGINNING IN COLUMN 3 .
X 40 SHOULD APPEAR BEGINNING IN COLUMN 16 .
X 41 SHOULD APPEAR BEGINNING IN COLUMN 29 .
X 42 SHOULD APPEAR BEGINNING IN COLUMN 42 .
X 43 SHOULD APPEAR BEGINNING IN COLUMN 55 .
X 44 SHOULD APPEAR BEGINNING IN COLUMN 68 .
X 45 SHOULD APPEAR BEGINNING IN COLUMN 9 .
X 46 SHOULD APPEAR BEGINNING IN COLUMN 22 .
X 47 SHOULD APPEAR BEGINNING IN COLUMN 35 .
X 48 SHOULD APPEAR BEGINNING IN COLUMN 48 .
X 49 SHOULD APPEAR BEGINNING IN COLUMN 61 .
X 50 SHOULD APPEAR BEGINNING IN COLUMN 2 .
X 51 SHOULD APPEAR BEGINNING IN COLUMN 15 .
X 52 SHOULD APPEAR BEGINNING IN COLUMN 28 .
X 53 SHOULD APPEAR BEGINNING IN COLUMN 41 .
X 54 SHOULD APPEAR BEGINNING IN COLUMN 54 .
X 55 SHOULD APPEAR BEGINNING IN COLUMN 67 .
X 56 SHOULD APPEAR BEGINNING IN COLUMN 8 .
X 57 SHOULD APPEAR BEGINNING IN COLUMN 21 .
X 58 SHOULD APPEAR BEGINNING IN COLUMN 34 .
X 59 SHOULD APPEAR BEGINNING IN COLUMN 47 .
X 60 SHOULD APPEAR BEGINNING IN COLUMN 60 .
X 61 SHOULD APPEAR BEGINNING IN COLUMN 1 .
X 62 SHOULD APPEAR BEGINNING IN COLUMN 14 .
X 63 SHOULD APPEAR BEGINNING IN COLUMN 27 .
X 64 SHOULD APPEAR BEGINNING IN COLUMN 40 .
X 65 SHOULD APPEAR BEGINNING IN COLUMN 53 .
X 66 SHOULD APPEAR BEGINNING IN COLUMN 66 .
X 67 SHOULD APPEAR BEGINNING IN COLUMN 7 .
X 68 SHOULD APPEAR BEGINNING IN COLUMN 20 .
X 69 SHOULD APPEAR BEGINNING IN COLUMN 33 .

```
X 70 SHOULD APPEAR BEGINNING IN COLUMN 46 .
X 71 SHOULD APPEAR BEGINNING IN COLUMN 59 .
X 72 SHOULD APPEAR BEGINNING IN COLUMN 72 .
X 73 SHOULD APPEAR BEGINNING IN COLUMN 13 .
X 74 SHOULD APPEAR BEGINNING IN COLUMN 26 .
X 75 SHOULD APPEAR BEGINNING IN COLUMN 39 .
X 76 SHOULD APPEAR BEGINNING IN COLUMN 52 .
X 77 SHOULD APPEAR BEGINNING IN COLUMN 65 .
X 78 SHOULD APPEAR BEGINNING IN COLUMN 6 .
X 79 SHOULD APPEAR BEGINNING IN COLUMN 19 .
X 80 SHOULD APPEAR BEGINNING IN COLUMN 32 .
X 81 SHOULD APPEAR BEGINNING IN COLUMN 45 .
X 82 SHOULD APPEAR BEGINNING IN COLUMN 58 .
X 83 SHOULD APPEAR BEGINNING IN COLUMN 71 .
X 84 SHOULD APPEAR BEGINNING IN COLUMN 12 .
X 85 SHOULD APPEAR BEGINNING IN COLUMN 25 .
X 86 SHOULD APPEAR BEGINNING IN COLUMN 38 .
X 87 SHOULD APPEAR BEGINNING IN COLUMN 51 .
X 88 SHOULD APPEAR BEGINNING IN COLUMN 64 .
X 89 SHOULD APPEAR BEGINNING IN COLUMN 5 .
X 90 SHOULD APPEAR BEGINNING IN COLUMN 18 .
X 91 SHOULD APPEAR BEGINNING IN COLUMN 31 .
X 92 SHOULD APPEAR BEGINNING IN COLUMN 44 .
X 93 SHOULD APPEAR BEGINNING IN COLUMN 57 .
X 94 SHOULD APPEAR BEGINNING IN COLUMN 70 .
X 95 SHOULD APPEAR BEGINNING IN COLUMN 11 .
X 96 SHOULD APPEAR BEGINNING IN COLUMN 24 .
X 97 SHOULD APPEAR BEGINNING IN COLUMN 37 .
X 98 SHOULD APPEAR BEGINNING IN COLUMN 50 .
X 99 SHOULD APPEAR BEGINNING IN COLUMN 63 .
X 100 SHOULD APPEAR BEGINNING IN COLUMN 4 .

                                END TEST.
```

SECTION 92.2: TABBING ASSIGNED STRINGS BEYOND THE CURRENT PRINT POSITION AS WELL AS BEYOND THE MARGIN.

```
                                BEGIN TEST.

0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000001111111111222222222233333333334444444444555555555566666666667777
1234567890123456789012345678901234567890123456789012345678901234567890123456789012

            A1
                        A2
                                    A3
                                                A4
                                                            A5
     A6
                  A7
                              A8
                                          A9
```

A10

A1

1

A12

A13

A14

A15

A16

A17

A18

A19

A20

A21

A22

A23

A24

A25

A26

A27

A28

A29

A30

A31

A32

A33

A34

A35

A36

A37

A38

A39

A40

A41

A42

A43

A44

A45

A46

A47

A48

A49

A50

A51

A52

A53

A54

A55

A56

A57

A58

A59

A60

A61

A62

A63

A64

A65

```
                                                                  A66
      A67
                A68
                          A69
                                    A70
                                              A71
                                                                       A
72
          A73
                    A74
                              A75
                                        A76
                                                  A77
     A78
               A79
                         A80
                                   A81
                                             A82
                                                                  A8
3
         A84
                   A85
                             A86
                                       A87
                                                 A88
    A89
              A90
                        A91
                                  A92
                                            A93
                                                                 A94
        A95
                  A96
                            A97
                                      A98
                                                A99
   A100
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000011111111112222222222333333333344444444445555555555666666666677
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

A1 SHOULD BEGIN IN COLUMN 13 .
A2 SHOULD BEGIN IN COLUMN 26 .
A3 SHOULD BEGIN IN COLUMN 39 .
A4 SHOULD BEGIN IN COLUMN 52 .
A5 SHOULD BEGIN IN COLUMN 65 .
A6 SHOULD BEGIN IN COLUMN 6 .
A7 SHOULD BEGIN IN COLUMN 19 .
A8 SHOULD BEGIN IN COLUMN 32 .
A9 SHOULD BEGIN IN COLUMN 45 .
A10 SHOULD BEGIN IN COLUMN 58 .
A11 SHOULD BEGIN IN COLUMN 71 .
A12 SHOULD BEGIN IN COLUMN 12 .
A13 SHOULD BEGIN IN COLUMN 25 .
A14 SHOULD BEGIN IN COLUMN 38 .
A15 SHOULD BEGIN IN COLUMN 51 .

```
A16 SHOULD BEGIN IN COLUMN 64 .
A17 SHOULD BEGIN IN COLUMN 5 .
A18 SHOULD BEGIN IN COLUMN 18 .
A19 SHOULD BEGIN IN COLUMN 31 .
A20 SHOULD BEGIN IN COLUMN 44 .
A21 SHOULD BEGIN IN COLUMN 57 .
A22 SHOULD BEGIN IN COLUMN 70 .
A23 SHOULD BEGIN IN COLUMN 11 .
A24 SHOULD BEGIN IN COLUMN 24 .
A25 SHOULD BEGIN IN COLUMN 37 .
A26 SHOULD BEGIN IN COLUMN 50 .
A27 SHOULD BEGIN IN COLUMN 63 .
A28 SHOULD BEGIN IN COLUMN 4 .
A29 SHOULD BEGIN IN COLUMN 17 .
A30 SHOULD BEGIN IN COLUMN 30 .
A31 SHOULD BEGIN IN COLUMN 43 .
A32 SHOULD BEGIN IN COLUMN 56 .
A33 SHOULD BEGIN IN COLUMN 69 .
A34 SHOULD BEGIN IN COLUMN 10 .
A35 SHOULD BEGIN IN COLUMN 23 .
A36 SHOULD BEGIN IN COLUMN 36 .
A37 SHOULD BEGIN IN COLUMN 49 .
A38 SHOULD BEGIN IN COLUMN 62 .
A39 SHOULD BEGIN IN COLUMN 3 .
A40 SHOULD BEGIN IN COLUMN 16 .
A41 SHOULD BEGIN IN COLUMN 29 .
A42 SHOULD BEGIN IN COLUMN 42 .
A43 SHOULD BEGIN IN COLUMN 55 .
A44 SHOULD BEGIN IN COLUMN 68 .
A45 SHOULD BEGIN IN COLUMN 9 .
A46 SHOULD BEGIN IN COLUMN 22 .
A47 SHOULD BEGIN IN COLUMN 35 .
A48 SHOULD BEGIN IN COLUMN 48 .
A49 SHOULD BEGIN IN COLUMN 61 .
A50 SHOULD BEGIN IN COLUMN 2 .
A51 SHOULD BEGIN IN COLUMN 15 .
A52 SHOULD BEGIN IN COLUMN 28 .
A53 SHOULD BEGIN IN COLUMN 41 .
A54 SHOULD BEGIN IN COLUMN 54 .
A55 SHOULD BEGIN IN COLUMN 67 .
A56 SHOULD BEGIN IN COLUMN 8 .
A57 SHOULD BEGIN IN COLUMN 21 .
A58 SHOULD BEGIN IN COLUMN 34 .
A59 SHOULD BEGIN IN COLUMN 47 .
A60 SHOULD BEGIN IN COLUMN 60 .
A61 SHOULD BEGIN IN COLUMN 1 .
A62 SHOULD BEGIN IN COLUMN 14 .
A63 SHOULD BEGIN IN COLUMN 27 .
A64 SHOULD BEGIN IN COLUMN 40 .
A65 SHOULD BEGIN IN COLUMN 53 .
A66 SHOULD BEGIN IN COLUMN 66 .
A67 SHOULD BEGIN IN COLUMN 7 .
A68 SHOULD BEGIN IN COLUMN 20 .
A69 SHOULD BEGIN IN COLUMN 33 .
A70 SHOULD BEGIN IN COLUMN 46 .
A71 SHOULD BEGIN IN COLUMN 59 .
A72 SHOULD BEGIN IN COLUMN 72 .
```

```
A73 SHOULD BEGIN IN COLUMN 13 .
A74 SHOULD BEGIN IN COLUMN 26 .
A75 SHOULD BEGIN IN COLUMN 39 .
A76 SHOULD BEGIN IN COLUMN 52 .
A77 SHOULD BEGIN IN COLUMN 65 .
A78 SHOULD BEGIN IN COLUMN 6 .
A79 SHOULD BEGIN IN COLUMN 19 .
A80 SHOULD BEGIN IN COLUMN 32 .
A81 SHOULD BEGIN IN COLUMN 45 .
A82 SHOULD BEGIN IN COLUMN 58 .
A83 SHOULD BEGIN IN COLUMN 71 .
A84 SHOULD BEGIN IN COLUMN 12 .
A85 SHOULD BEGIN IN COLUMN 25 .
A86 SHOULD BEGIN IN COLUMN 38 .
A87 SHOULD BEGIN IN COLUMN 51 .
A88 SHOULD BEGIN IN COLUMN 64 .
A89 SHOULD BEGIN IN COLUMN 5 .
A90 SHOULD BEGIN IN COLUMN 18 .
A91 SHOULD BEGIN IN COLUMN 31 .
A92 SHOULD BEGIN IN COLUMN 44 .
A93 SHOULD BEGIN IN COLUMN 57 .
A94 SHOULD BEGIN IN COLUMN 70 .
A95 SHOULD BEGIN IN COLUMN 11 .
A96 SHOULD BEGIN IN COLUMN 24 .
A97 SHOULD BEGIN IN COLUMN 37 .
A98 SHOULD BEGIN IN COLUMN 50 .
A99 SHOULD BEGIN IN COLUMN 63 .
A100 SHOULD BEGIN IN COLUMN 4 .

                                    END TEST.
```

93.0 EXCEPTION TEST - STRING OVERFLOW

A preliminary test of acceptable assigned string lengths was done in section 57.0. At that time strings of length 19, 20, 30, 40, 50, and 58 characters were used. Since program lines had to be restricted to 72 characters, inordinately long strings, used as program constants, could not be tested. However, with the INPUT statement longer strings can be introduced. The present test is based upon the assumption that a BASIC processor will have an input buffer of some maximum length, which may be unknown to the user. But, if the user enters a sufficiently long string, an overflow is inevitable. In order to execute this test one must assume that characters can be continuously typed on a terminal and that a carriage return or line feed character is not introduced into the string unless they are specifically entered by the user (say by a RETURN key). The user begins the test by entering 18 characters. All succeeding entries will be in multiples of 36. The user will then be asked to enter a sequence of strings of increasing length. If no overflow is encountered at an entry the user will be asked to enter a longer string. When a string overflow is finally encountered, the test system must provide a message to the user indicating the overflow and requesting input reentry (see section 13.5 of BSR X3.60). The user should then type STOP since the system would have responded properly to the string overflow. The authors have experienced at least one system that allowed a string entry of more than 145 characters before an overflow message was encountered.

```
********************
* PROGRAM FILE 93 *
********************
```

```
10 PRINT "PROGRAM FILE 93"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "                                SECTION 93.0"
60 PRINT
70 PRINT "CONVERSION OF A STRING INPUT DATUM CAUSES A STRING OVERFLOW"
90 PRINT
100 PRINT "                                BEGIN TEST"
120 PRINT
130 PRINT "THERE ARE TWO CRITERIA THAT MUST BE MET TO PASS THIS TEST. "
135 PRINT "FIRST, IF A STRING OVERFLOW IS ENCOUNTERED THEN THE USER "
140 PRINT "MUST BE ALLOWED TO REENTER THE STRING INPUT. SECOND, THE "
145 PRINT "FIRST STRING INPUT OF 18 CHARACTERS MUST BE ACCEPTED "
150 PRINT "WITHOUT ANY DIAGNOSTIC."
160 PRINT
170 PRINT "UPON ENCOUNTERING AN OVERFLOW AND A REQUEST TO REENTER DATA"
175 PRINT "THE USER SHOULD TYPE STOP, WHICH TERMINATES THE PROGRAM."
180 PRINT
```

```
190 PRINT "UPON INPUT PROMPT ENTER THE 18 CHARACTERS SHOWN BELOW."
210 PRINT
220 PRINT "XXXXXXXXXXXXXXXXXX"
230 INPUT A$
240 IF A$<>"STOP" THEN 270
250 PRINT "THIS SYSTEM HAS FAILED TO ACCEPT AN 18 CHARACTER STRING."
260 GO TO 470
270 PRINT
280 PRINT "FURTHER INPUT STRINGS WILL BE IN MULTIPLES OF 36 CHARACTER."
285 LET Q = 0
290 PRINT
320 PRINT "PLEASE DUPLICATE THE CHARACTER STRINGS DISPLAYED BEFORE EACH"
325 PRINT "INPUT PROMPT."
330 FOR I = 1+Q*100 TO 100+Q*100
340 PRINT
345 LET L = 36*I
350 PRINT "NEW INPUT STRING:"; L ; " CHARACTERS"
360 FOR J = 1 TO L
370 PRINT "X";
380 NEXT J
385 PRINT
390 INPUT A$
400 IF A$<>"STOP" THEN 430
410 PRINT "IF STOP WAS TYPED DUE TO AN OVERFLOW MESSAGE THEN TEST"
415 PRINT "HAS PASSED WITH THIS MESSAGE."
420 GO TO 470
430 NEXT I
440 PRINT
450 PRINT "NO OVERFLOWS UP TO THIS POINT. THE USER MUST CONTINUE."
460 PRINT "IF STOP IS TYPED BEFORE AN OVERFLOW THEN THE TEST IS"
465 PRINT "INVALID AND MUST BE RUN AGAIN."
467 LET Q = Q + 1
468 GO TO 320
470 END
```

```
******************
* SAMPLE OUTPUT *
******************
```

PROGRAM FILE 93

SECTION 93.0

CONVERSION OF A STRING INPUT DATUM CAUSES A STRING OVERFLOW

BEGIN TEST

```
THERE ARE TWO CRITERIA THAT MUST BE MET TO PASS THIS TEST.
FIRST, IF A STRING OVERFLOW IS ENCOUNTERED THEN THE USER
MUST BE ALLOWED TO REENTER THE STRING INPUT. SECOND, THE
FIRST STRING INPUT OF 18 CHARACTERS MUST BE ACCEPTED
WITHOUT ANY DIAGNOSTIC.

UPON ENCOUNTERING AN OVERFLOW AND A REQUEST TO REENTER DATA
THE USER SHOULD TYPE STOP, WHICH TERMINATES THE PROGRAM.

UPON INPUT PROMPT ENTER THE 18 CHARACTERS SHOWN BELOW.

XXXXXXXXXXXXXXXXX
?XXXXXXXXXXXXXXXXXX

FURTHER INPUT STRINGS WILL BE IN MULTIPLES OF 36 CHARACTER.

PLEASE DUPLICATE THE CHARACTER STRINGS DISPLAYED BEFORE EACH
INPUT PROMPT.

NEW INPUT STRING: 36  CHARACTERS
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

NEW INPUT STRING: 72  CHARACTERS
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X

NEW INPUT STRING: 108  CHARACTERS
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

NEW INPUT STRING: 144  CHARACTERS
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X

NEW INPUT STRING: 180  CHARACTERS
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
?INPUT STRING TOO LONG. PLEASE REENTER.
?STOP
```

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS IR 78-1420-3	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

NBS Minimal BASIC Test Programs - Version 1
User's Manual
Volume 3 - Control Statements, Data Structure, Program Input

5. Publication Date
January 1978

6. Performing Organization Code

7. AUTHOR(S)
David E. Gilsinn and Charles L. Sheppard

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

10. Project/Task/Work Unit No.
6401121

11. Contract/Grant No.

12. Sponsoring Organization Name and Complete Address (Street, City, State, ZIP)

National Bureau of Standards

13. Type of Report & Period Covered
Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

This volume is the third of four volumes that comprise the user's guide to the NBS Minimal BASIC Test Programs. The programs test whether a BASIC processor accepts the syntactical forms and produces semantically meaningful results according to the specifications given in BSR X3.60 Proposed American National Standard for Minimal BASIC. The object of this volume is to complete the testing of the control structures, introduce new data structures, and test the user interactive capability of the language. There are sixty individual programs in this volume that cover looping structures, array variables, exception tests, subroutines, multiway branch structures, data declarations and interactive data inputs. The entire set of programs is available on tape.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

BASIC; BASIC standard; BASIC validation; compiler validation; computer programming language; computer standards.

18. AVAILABILITY

- [ ] Unlimited
- [x] For Official Distribution. Do Not Release to NTIS
- [ ] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13
- [ ] Order From National Technical Information Service (NTIS) Springfield, Virginia 22151

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PAGES
UNCLASSIFIED	239
20. SECURITY CLASS (THIS PAGE)	**22. Price**
UNCLASSIFIED	

USCOMM-DC 29042-P74