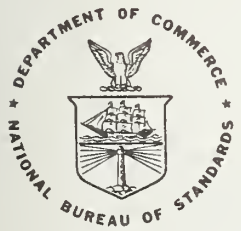
NBSIR 78-1420-4

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

Volume 4 - Mathematical and User Defined Functions, Compound Expressions

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 78-1420-4

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

Volume 4 - Mathematical and User Defined Functions, Compound Expressions

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 fourth of four volumes that comprise the user's guide to the NBS Minimal BASIC test programs. They 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 programs in this volume include: mathematical function tests; exception tests for some of the functions; a statistical test of the uniformness of the random number generator; user defined function tests; syntax tests on the functions; tests using compound arithmetic expressions and certain semantic tests on arrays and variable initialization. The entire set of test programs is available on magnetic tape.

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

Table of Contents

0.0 INTRODUCTION

This is the final volume of a four volume set. The programs documented in these volumes form the NBS test system. They are intended for use as validation tools to test the conformance of a BASIC computer language processor to the American National Standard for Minimal BASIC X3.60. The individual routines in this volume include: mathematical function tests; exception tests for some of the functions; statistical tests for the random number generator; user-defined function tests; syntax tests on the functions; and tests using compound expressions as (1) arguments for functions, (2) parameters in control statements, and (3) print items. The final programs include semantic error diagnostic tests. In order to understand the context of this volume the reader should be familiar with the first three volumes to this set and the American National Standard for Minimal BASIC, BSR X3.60.

Each section contains short statements of the objectives of the test program associated with the section. Many of the sections have subsections that correspond to subtests within the main test program for that section. After each set of descriptive paragraphs, a listing of the program is given, and finally a sample output is reproduced.

Sections 94 through 107 introduce the trigonometric and other mathematical functions by sampling the accuracy maintained by each. The Minimal BASIC standard does not specify the accuracy requirements for the implementation supplied mathematical functions. However, a user needs some assurance of accuracy in order to have faith in his calculations. Therefore, a method to detect and flag any "computed deviations" from a function's "true results", to within less than one unit in the sixth significant digit of the function's evaluations of test arguments, is included. Each function test routine contains a sample of arguments and true evaluations rounded to six significant digits. The sample arguments essentially cover the range of permitted numbers of absolute value from 1E-38 to 1E+38. There is a detailed discussion of the numerical comparison procedures used for each of the functions given in section 94.0. The same procedure is used with each of the other mathematical functions.

A pseudo-random number generator is accessible to BASIC programs. Sections 108 to 110 introduce the implementation-supplied random number generator RND. A call to RND generates the next pseudo-random number in an implementation-supplied sequence of numbers, uniformly distributed in the interval 0<= RND < 1. When the statement RANDOMIZE is used in a program the sequence is different for each execution of the program. A statistical test program is included in section 110.0. It assesses the uniformity of the sequence of pseudo-random numbers. The test used is a goodness-of-fit test. In particular the test performs thirty chi-square tests, and then performs a goodness-of-fit test of the resultant chi-square values to the cumulative chi-square distribution. These two steps give information both on the local and the global behavior of the pseudo-random number sequence.

Tests for user-defined functions are given in sections 111 to 116. These functions can be defined by users with and without parameters. Several programs test defined functions. Some of the program outputs call for diagnostics because the defined-functions used are semantically non-meaningful.

The second half of this volume includes: syntax tests for the mathematical functions and a set of tests that emphasize various uses of mathematical functions and arrays. In these tests compound expressions are also substituted for the parameter at call time. Compound expressions are also used as expressions in many of the control statements.

It should be reiterated that the tests are concerned not only with the language conformance to the ANSI BSR X3.60 specifications, but also with the implementation conformance of the language since the user ultimately confronts the implementation. In general the user expects the language to be a useful tool to solve a problem. In order to solve problems, however, the implementation of the language must perform operations that are meaningful to the user. This is why a great deal of attention has been paid to implementation questions in this set of tests.

94.0 THE SQR FUNCTION

The objective of this test is to introduce the use of the SQR function by performing a short evaluation of the accuracy maintained by the implementation-supplied function for the host system (see section 8 of BSR X3.60. These tests are not intended to be exhaustive, but are intended to show a broad view of what range of numbers the user can expect adequate function performance. For a discussion of more thorough evaluation techniques the reader is referred to C. T. Fike, Computer Evaluation of Mathematical Functions, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1968) or J. F. Hart, et. al., Computer Approximations, John Wiley and Sons, Inc., New York (1968).

The test first compares the evaluation of the SQR function on a set of arguments with a known set of results, and then computes the relative error of the system-generated result with respect to the known result. The known-values were computed by an extended precision routine and rounded to six significant digits. All of this data is displayed in four columns. The first column is labeled ARGUMENT, the second column is labeled TEST VALUE, the third column is labeled SYSTEM VALUE, and the fourth column is labeled RELATIVE ERROR. At the end, the error range is displayed.

The "relative error" concept is used because it allows one to detect a deviation of one digit in the last significant place, which in this case is the sixth significant digit. This is done, in fact, without the need to know an implementation-defined machine "infinitesimal". The argument is as follows: First suppose that a floating point number y is represented in the normalized form y = mEa where Ea, of course, represents ten to the power a and we suppose that 1/10 <= ABS(m) < 1. Let z = nEa be an approximation to y. Then y and z agree within six significant digits, assuming rounding arithmetic, provided that ABS(m-n) <= 0.5E-6. The relative error between y and z, assuming y is not zero, can be written as ABS((z-y)/y) = ABS((n-m)/m) <= (0.5E-6)/ABS(m). But ABS(y) = ABS(m) * 1Ea so that 1/ABS(m) = 1Ea/ABS(y). Therefore we have for the case y <> 0 that ABS((z-y)/y) <= (0.5E-6) * (P/Y1) where we have set P = 1Ea and Y1 = ABS(y). This gives us a formula to detect any deviation including the last significant digit, provided we determine P. This is fairly simple. Instead of storing this value for each y used we can compute P by selecting that power 10 such that 10^(a-1) <= y < 10^a. Then let P = 10^a = 1Ea. We note that in order not to overflow, scaling factors S1 and S2 have been used in the function test codes, in order to maintain numbers within manageable size. Finally, in order to handle the case y = 0 we have to distinguish two possibilities: The case z = 0 and z <> 0. If z = 0 then there is no error and we define the relative error to be 0. If z <> 0 then we exchange the roles of y and z.

The user need only look for asterisk (*) flags to the right of the relative error column. If there are none, even though positive relative errors show, the test has "passed". This is an informative test, and is intended to perform two tasks. First, it is intended as a program to demonstrate the pure language capability of the implementation, with respect to accessing a supplied function. However, a user might be somewhat upset if SQR(25) <> 5. Therefore, secondly, the test seeks to sample, albeit not exhaustively, the accuracy of the supplied function. ANSI Minimal BASIC does not specify the accuracy of the supplied function, but it does recommend that arithmetic operations attempt to maintain six significant places. Therefore this test demands accuracy within six significant digits. The user should

pay attention, on some systems, to any deterioration in accuracy of some of the functions that occur at large arguments; points at which the function becomes unbounded; and points at which the function is near zero. For designers there are references that can be consulted in order to develop high quality function subroutines. The fundamental reference is John F. Hart, et al., Computer Approximations, John Wiley & Sons, Inc., New York (1968).

```
********************
* PROGRAM FILE 94 *
********************
```

```
0010 PRINT "PROGRAM FILE 94"
0020 PRINT
0030 PRINT
0040 PRINT
0080 PRINT
0090 PRINT "                              SECTION 94.0"
0100 PRINT
0110 PRINT "                         SQUARE ROOT FUNCTION (SQR)"
0120 PRINT
0130 PRINT "                                BEGIN TEST"
0140 PRINT
0150 DATA  .00000     ,  .000000     ,  .10000E-34 ,  .316228E-17
0160 DATA  .10000E-15 ,  .100000E-07 ,  .10000E-07 ,  .100000E-03
0170 DATA  .10000E-03 ,  .100000E-01 ,  .10000E-01 ,  .100000E+00
0180 DATA  .98491E-01 ,  .313833E+00 ,  .20000E+00 ,  .447214E+00
0190 DATA  .26795E+00 ,  .517639E+00 ,  .50000E+00 ,  .707107E+00
0200 DATA  .10000E+01 ,  .100000E+01 ,  .30000E+01 ,  .173205E+01
0210 DATA  .50000E+01 ,  .223607E+01 ,  .11000E+02 ,  .331662E+01
0220 DATA  .25000E+02 ,  .500000E+01 ,  .75000E+02 ,  .866025E+01
0230 DATA  .10000E+11 ,  .100000E+06 ,  .10000E+36 ,  .316228E+18
0240 DIM E(20)
0250 LET L = 0
0260 LET H = 0
0270 LET M = 0
0280 LET V = 0
0290 PRINT " "," "," ","  RELATIVE "
0300 PRINT "   ARGUMENT", " TEST VALUE", " SYSTEM VALUE", " ERROR"
0310 FOR I = 1 TO 18
0315 LET S1 = 1
0320 READ X, Y
0330 LET A$ = " "
0340 LET Z = SQR(X)
0370 IF ABS(Y) >= 1E-38 THEN 377
0371 IF ABS(Z) >= 1E-38 THEN 374
0372 LET E(I) = 0.0
0373 GO TO 420
0374 LET T1 = Y
0375 LET Y = Z
```

```
0376 LET Z = T1
0377 REM TEST Y FOR SCALING
0378 IF ABS(Y) <= 1E30 THEN 380
0379 LET S1 = 1E-8
0380 IF ABS(Y) >= 1E-30 THEN 382
0381 LET S1 = 1E8
0382 LET E(I) = ((S1*Z) - (S1*Y))/(S1*Y)
0383 LET E(I) = ABS(E(I))
0384 LET A1 = E(I)
0385 LET Y2 = Y
0386 GOSUB 9000
0387 IF ABS(A1) < C THEN 420
0410 LET A$ = "****"
0420 PRINT X,Y,Z,E(I);A$
0430 NEXT I
0440 PRINT
0450 PRINT "                    EVALUATION OF THE ABSOLUTE RELATIVE ";
0460 PRINT "ERRORS"
0470 PRINT
0480 FOR I = 1 TO 18
0490 IF H > E(I) THEN 520
0500 LET H = E(I)
0510 GO TO 540
0520 IF L <= E(I) THEN 540
0530 LET L = E(I)
0540 NEXT I
0700 PRINT
0710 PRINT "        LOWEST ERROR =";L;"  HIGHEST ERROR =";H
0720 PRINT
0810 PRINT "                                END TEST"
0820 PRINT
0850 GO TO 9400
9000 IF 1 <= ABS(Y2) THEN 9101
9001 LET S2 = 1
9003 IF ABS(Y2) > 1E-37 THEN 9010
9005 LET P = 1E-37
9007 LET S2 = 1E10
9009 GO TO 9200
9010 LET P = 1
9020 LET P1 = P/10
9030 IF P1 <= ABS(Y2) THEN 9200
9040 LET P = P/10
9050 GO TO 9020
9101 LET S2 = 1
9105 IF ABS(Y2) <= 1E37 THEN 9110
9106 LET P = 1E38
9107 LET S2 = 1E-10
9109 GO TO 9200
9110 LET P = 10
9120 IF ABS(Y2) < P THEN 9200
9130 LET P = P*10
9140 GO TO 9120
9200 LET Y1 = ABS(Y2)
9210 LET C = (.5E-6)*((P*S2)/(Y1*S2))
9300 RETURN
9400 END
```

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

PROGRAM FILE 94

SECTION 94.0

SQUARE ROOT FUNCTION (SQR)

BEGIN TEST

ARGUMENT	TEST VALUE	SYSTEM VALUE	RELATIVE ERROR
0	0	0	0
1.00000E-35	3.16228E-18	3.16228E-18	0
1.00000E-16	1.00000E-8	1.00000E-8	0
1.00000E-8	.0001	.0001	0
.0001	.01	.01	0
.01	.1	.1	0
.098491	.313833	.313833	0
.2	.447214	.447214	0
.26795	.517639	.517639	0
.5	.707107	.707107	0
1	1	1	0
3	1.73205	1.73205	0
5	2.23607	2.23607	0
11	3.31662	3.31662	0
25	5	5	0
75	8.66025	8.66025	0
1.00000E+10	100000	100000	0
1.00000E+35	3.16228E+17	3.16228E+17	0

EVALUATION OF THE ABSOLUTE RELATIVE ERRORS

LOWEST ERROR = 0 HIGHEST ERROR = 0

END TEST

95.0 EXCEPTION TEST FOR THE SQR FUNCTION

The objective of this test is to verify that the implementation recognizes the use of a negative numerical argument for the SQR function as an exception (see section 8.5 of BSR X3.60). Upon recognition of the exception, it should be reported and the program execution suspended. Some implementations return the square root of the absolute value and continue the program. If the latter occurs, the test fails since the ANSI Minimal BASIC Standard does not prescribe any specified procedures to handle this exception. This test allows the argument of the SQR function to be assigned a negative value, as found in lines 240 and 250 in program file 95. Output should contain some form of implementation-defined diagnostic concerning the error.

```
********************
* PROGRAM FILE 95 *
********************
```

```
0010 PRINT "PROGRAM FILE 95"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "      SECTION 95.0: EXCEPTION TEST FOR THE SQR FUNCTION."
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0140 PRINT "THIS SYSTEM RECOGNIZES AN ARGUMENT OF THE SQR FUNCTION WITH"
0150 PRINT "A NEGATIVE VALUE AS AN EXCEPTION. IT SHOULD BE REPORTED"
0170 PRINT "IF SUCH A RECOGNITION SHOULD OCCUR, CONSIDER SYSTEM TO HAVE"
0180 PRINT "PASSED TEST."
0190 PRINT
0200 PRINT
0210 PRINT
0220 PRINT "                                BEGIN TEST."
0230 PRINT
0240 LET X=-64
0250 LET Y=SQR(X)
0260 PRINT "IF A NUMERICAL VALUE IS PRINTED BELOW WITHOUT A DIAGNOSTIC"
0270 PRINT "THE SYSTEM FAILED THE TEST."
0280 PRINT Y
0290 PRINT
0300 PRINT "                                  END TEST."
0310 PRINT
0320 END
```

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

PROGRAM FILE 95

SECTION 95.0: EXCEPTION TEST FOR THE SQR FUNCTION.

THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER THIS SYSTEM RECOGNIZES AN ARGUMENT OF THE SQR FUNCTION WITH A NEGATIVE VALUE AS AN EXCEPTION. IT SHOULD BE REPORTED IF SUCH A RECOGNITION SHOULD OCCUR, CONSIDER SYSTEM TO HAVE PASSED TEST.

BEGIN TEST.

?SQRT OF NEGATIVE NUMBER IN LINE 250

96.0 THE ATN FUNCTION

The objective of this test is to introduce the use of the ATN function, by sampling the accuracy maintained by the implementation-supplied function (see section 8 of BSR X3.60). The ATN function returns the angle in radians whose TAN is the argument of ATN. The range of the ATN function varies from -(pi/2) to (pi/2). The actual test, except for using the ATN function instead of the SQR function, is the same in structure and output as that of section 94.0.

```
********************
* PROGRAM FILE 96 *
********************
```

```
0010 PRINT "PROGRAM FILE 96"
0020 PRINT
0030 PRINT
0040 PRINT
0080 PRINT
0090 PRINT "                              SECTION 96.0"
0100 PRINT
0110 PRINT "                         ARCTANGENT FUNCTION (ATN)"
0120 PRINT
0130 PRINT "                                    BEGIN TEST"
0140 DATA -.10000E+26 , -.157080E+01 , -.10000E+03 , -.156080E+01
0150 DATA -.85000E+02 , -.155903E+01 , -.50000E+02 , -.155080E+01
0160 DATA -.16000E+02 , -.150838E+01 , -.80000E+01 , -.144644E+01
0170 DATA -.40000E+01 , -.132582E+01 , -.20000E+01 , -.110715E+01
0180 DATA -.80000     , -.674741     , -.41421     , -.392696
0190 DATA -.30000     , -.291457     , -.19891     , -.196347
0200 DATA -.10000     , -.996687E-01 , -.10000E-02 , -.100000E-02
0210 DATA -.10000E-04 , -.100000E-04 , -.10000E-10 , -.100000E-10
0220 DATA -.10000E-24 , -.100000E-24 ,  .00000     ,  .000000
0230 DATA  .10000E-34 ,  .100000E-34 ,  .10000E-15 ,  .100000E-15
0240 DATA  .10000E-07 ,  .100000E-07 ,  .10000E-03 ,  .100000E-03
0250 DATA  .10000E-01 ,  .999967E-02 ,  .98491E-01 ,  .981744E-01
0260 DATA  .20000     ,  .197396     ,  .26795     ,  .261800
0270 DATA  .50000     ,  .463648     ,  .10000E+01 ,  .785398
0280 DATA  .30000E+01 ,  .124905E+01 ,  .50000E+01 ,  .137340E+01
0290 DATA  .11000E+02 ,  .148014E+01 ,  .25000E+02 ,  .153082E+01
0300 DATA  .75000E+02 ,  .155746E+01 ,  .10000E+11 ,  .157080E+01
0310 DATA  .10000E+36 ,  .157080E+01
0320 DIM E(35)
0330 LET L = 0
0340 LET H = 0
0350 LET M = 0
0360 LET V = 0
0370 PRINT " ", " ", " ","RELATIVE"
```

```
0380 PRINT "   ARGUMENT"," TEST VALUE"," SYSTEM VALUE","  ERROR"
0390 FOR I = 1 TO 35
0395 LET S1 = 1
0400 READ X,Y
0410 LET A$ = " "
0420 LET Z = ATN(X)
0425 IF ABS(Y) >= 1E-38 THEN 432
0426 IF ABS(Z) >= 1E-38 THEN 429
0427 LET E(I) = 0.0
0428 GO TO 500
0429 LET T1 = Y
0430 LET Y = Z
0431 LET Z = T1
0432 REM TEST Y FOR SCALING
0433 IF ABS(Y) <= 1E30 THEN 435
0434 LET S1 = 1E-8
0435 IF ABS(Y) >= 1E-30 THEN 437
0436 LET S1 = 1E8
0437 LET E(I) = ((S1*Z) - (S1*Y))/(S1*Y)
0438 LET E(I) = ABS(E(I))
0439 LET A1 = E(I)
0440 LET Y2 = Y
0441 GOSUB 9000
0442 IF ABS(A1) < C THEN 500
0490 LET A$ = "****"
0500 PRINT X,Y,Z,E(I);A$
0510 NEXT I
0520 PRINT
0530 PRINT "                    EVALUATION OF THE ABSOLUTE RELATIVE";
0540 PRINT " ERRORS"
0550 PRINT
0560 PRINT
0570 FOR I = 1 TO 35
0580 IF H > E(I) THEN 610
0590 LET H = E(I)
0600 GO TO 630
0610 IF L <= E(I) THEN 630
0620 LET L = E(I)
0630 NEXT I
0740 PRINT
0750 PRINT "         LOWEST ERROR =";L;"  HIGHEST ERROR =";H
0760 PRINT
0850 PRINT "                                   END TEST"
0860 PRINT
0870 GO TO 9400
9000 IF 1 <= ABS(Y2) THEN 9101
9001 LET S2 = 1
9003 IF ABS(Y2) > 1E-37 THEN 9010
9005 LET P = 1E-37
9007 LET S2 = 1E10
9009 GO TO 9200
9010 LET P = 1
9020 LET P1 = P/10
9030 IF P1 <= ABS(Y2) THEN 9200
9040 LET P = P/10
9050 GO TO 9020
9101 LET S2 = 1
```

```
9105 IF ABS(Y2) <= 1E37 THEN 9110
9106 LET P = 1E38
9107 LET S2 = 1E-10
9109 GO TO 9200
9110 LET P = 10
9120 IF ABS(Y2) < P THEN 9200
9130 LET P = P*10
9140 GO TO 9120
9200 LET Y1 = ABS(Y2)
9210 LET C = (.5E-6)*((P*S2)/(Y1*S2))
9300 RETURN
9400 END
```

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

PROGRAM FILE 96

SECTION 96.0

ARCTANGENT FUNCTION (ATN)

BEGIN TEST

ARGUMENT	TEST VALUE	SYSTEM VALUE	RELATIVE ERROR
-1.00000E+25	-1.5708	-1.5708	0
-100	-1.5608	-1.5608	0
-85	-1.55903	-1.55903	0
-50	-1.5508	-1.5508	0
-16	-1.50838	-1.50838	0
-8	-1.44644	-1.44644	0
-4	-1.32582	-1.32582	0
-2	-1.10715	-1.10715	0
-.8	-.674741	-.674741	0
-.41421	-.392696	-.392696	0
-.3	-.291457	-.291457	0
-.19891	-.196347	-.196347	0
-.1	-9.96687E-2	-9.96687E-2	0
-.001	-.001	-.001	0
-.00001	-.00001	-.00001	0
-1.00000E-11	-1.00000E-11	-1.00000E-11	0
-1.00000E-25	-1.00000E-25	-1.00000E-25	0
0	0	0	0
1.00000E-35	1.00000E-35	1.00000E-35	0
1.00000E-16	1.00000E-16	1.00000E-16	0

1.00000E-8	1.00000E-8	1.00000E-8	0
.0001	.0001	.0001	0
.01	9.99967E-3	9.99967E-3	0
.098491	9.81744E-2	9.81744E-2	0
.2	.197396	.197396	0
.26795	.2618	.2618	0
.5	.463648	.463648	0
1	.785398	.785398	0
3	1.24905	1.24905	0
5	1.3734	1.3734	0
11	1.48014	1.48014	0
25	1.53082	1.53082	0
75	1.55746	1.55746	0
1.00000E+10	1.5708	1.5708	0
1.00000E+35	1.5708	1.5708	0

EVALUATION OF THE ABSOLUTE RELATIVE ERRORS

LOWEST ERROR = 0 HIGHEST ERROR = 0

END TEST

97.0 THE COS FUNCTION

The objective of this test is to introduce the use of the COS function by sampling the accuracy maintained by the implementation-supplied function (see section 8 of BSR X3.60). The COS function returns the cosine of an argument entered as an argument in radians. This test, except for the use of the COS function instead of the SQR function, is the same as section 94..

```
**********
* PROGRAM FILE 97 *
**********
```

```
0010 PRINT "PROGRAM FILE 97"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                              SECTION 97.0"
0100 PRINT
0110 PRINT "                          COSINE FUNCTION (COS)"
0120 PRINT
0130 PRINT "                                BEGIN TEST"
0140 DATA -.10000E+05 , -.952155E+00 , -.10000E+03 ,  .862319E+00
0150 DATA -.50000E+02 ,  .964966E+00 , -.16000E+02 , -.957659E+00
0160 DATA -.80000E+01 , -.145500E+00 , -.40000E+01 , -.653644E+00
0170 DATA -.20000E+01 , -.416147E+00 , -.50000E+00 ,  .877583E+00
0180 DATA -.10000E+00 ,  .995004E+00 , -.10000E-02 ,  .100000E+01
0190 DATA -.10000E-04 ,  .100000E+01 , -.10000E-10 ,  .100000E+01
0200 DATA -.10000E-24 ,  .100000E+01 ,  .00000     ,  .100000E+01
0210 DATA  .10000E-34 ,  .100000E+01 ,  .10000E-15 ,  .100000E+01
0220 DATA  .10000E-07 ,  .100000E+01 ,  .10000E-03 ,  .100000E+01
0230 DATA  .10000E-01 ,  .999950E+00 ,  .25000E+00 ,  .968912E+00
0240 DATA  .26179E+00 ,  .965928E+00 ,  .52360E+00 ,  .866025E+00
0250 DATA  .75000E+00 ,  .731689E+00 ,  .78539E+00 ,  .707113E+00
0260 DATA  .10000E+01 ,  .540302E+00 ,  .10472E+01 ,  .499998E+00
0270 DATA  .13089E+01 ,  .258913E+00 ,  .18325E+01 , -.258727E+00
0280 DATA  .20944E+01 , -.500004E+00 ,  .23561E+01 , -.707040E+00
0290 DATA  .26180E+01 , -.866028E+00 ,  .28797E+01 , -.965902E+00
0300 DATA  .30000E+01 , -.989992E+00 ,  .31416E+01 , -.100000E+01
0310 DATA  .34033E+01 , -.965950E+00 ,  .36652E+01 , -.866021E+00
0320 DATA  .39279E+01 , -.706464E+00 ,  .41888E+01 , -.499992E+00
0330 DATA  .44505E+01 , -.258906E+00 ,  .49741E+01 ,  .258734E+00
0340 DATA  .50000E+01 ,  .283662E+00 ,  .52360E+01 ,  .500011E+00
0350 DATA  .54977E+01 ,  .707045E+00 ,  .57596E+01 ,  .866032E+00
0360 DATA  .60213E+01 ,  .965904E+00 ,  .62832E+01 ,  .100000E+01
0370 DATA  .11000E+02 ,  .442570E-02 ,  .25000E+02 ,  .991203E+00
0380 DATA  .75000E+02 ,  .921751E+00 ,  .10000E+04 ,  .562379E+00
0390 DATA  .10000E+06 , -.999361E+00
0400 DIM E(51)
```

```
0410 LET L = 0
0420 LET H = 0
0430 LET M = 0
0440 LET V = 0
0450 PRINT " ", " "," ", "RELATIVE"
0460 PRINT "   ARGUMENT", "TEST VALUE", " SYSTEM VALUE", "   ERROR"
0470 FOR I = 1 TO 51
0475 LET S1 = 1
0480 READ X,Y
0490 LET A$ = " "
0500 LET Z = COS(X)
0501 IF ABS(Y) >= 1E-38 THEN 508
0502 IF ABS(Z) >= 1E-38 THEN 505
0503 LET E(I) = 0.0
0504 GO TO 580
0505 LET T1 = Y
0506 LET Y = Z
0507 LET Z = T1
0508 REM TEST Y FOR SCALING
0509 IF ABS(Y) <= 1E30 THEN 511
0510 LET S1 = 1E-8
0511 IF ABS(Y) >= 1E-30 THEN 513
0512 LET S1 = 1E8
0513 LET E(I) = ((S1*Z) - (S1*Y))/(S1*Y)
0514 LET E(I) = ABS(E(I))
0515 LET A1 = E(I)
0516 LET Y2 = Y
0517 GOSUB 9000
0518 IF ABS(A1) < C THEN 580
0570 LET A$ = "****"
0580 PRINT X,Y,Z,E(I);A$
0590 NEXT I
0600 PRINT
0610 PRINT "                    EVALUATION OF THE ABSOLUTE RELATIVE";
0620 PRINT " ERRORS"
0630 PRINT
0640 PRINT
0650 FOR I = 1 TO 51
0660 IF H > E(I) THEN 690
0670 LET H = E(I)
0680 GO TO 710
0690 IF L<= E(I) THEN 710
0700 LET L = E(I)
0710 NEXT I
0820 PRINT
0830 PRINT "        LOWEST ERROR =";L;"  HIGHEST ERROR =";H
0840 PRINT
0930 PRINT "                                   END TEST"
0940 PRINT
1000 GO TO 9400
9000 IF 1 <= ABS(Y2) THEN 9101
9001 LET S2 = 1
9003 IF ABS(Y2) > 1E-37 THEN 9010
9005 LET P = 1E-37
9007 LET S2 = 1E10
9009 GO TO 9200
9010 LET P = 1
```

```
9020 LET P1 = P/10
9030 IF P1 <= ABS(Y2) THEN 9200
9040 LET P = P/10
9050 GO TO 9020
9101 LET S2 = 1
9105 IF ABS(Y2) <= 1E37 THEN 9110
9106 LET P = 1E38
9107 LET S2 = 1E-10
9109 GO TO 9200
9110 LET P = 10
9120 IF ABS(Y2) < P THEN 9200
9130 LET P = P*10
9140 GO TO 9120
9200 LET Y1 = ABS(Y2)
9210 LET C = (.5E-6)*((P*S2)/(Y1*S2))
9300 RETURN
9400 END
```

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

PROGRAM FILE 97

SECTION 97.0

COSINE FUNCTION (COS)

BEGIN TEST

ARGUMENT	TEST VALUE	SYSTEM VALUE	RELATIVE ERROR
-10000	-.952155	-.952155	0
-100	.862319	.862319	0
-50	.964966	.964966	0
-16	-.957659	-.95766	0
-8	-.1455	-.1455	0
-4	-.653644	-.653644	0
-2	-.416147	-.416147	0
-.5	.877583	.877583	0
-.1	.995004	.995004	0
-.001	1	1	0
-.00001	1	1	0
-1.00000E-11	1	1	0
-1.00000E-25	1	1	0
0	1	1	0
1.00000E-35	1	1	0
1.00000E-16	1	1	0

1.00000E-8	1	1	0
.0001	1	1	0
.01	.99995	.99995	0
.25	.968912	.968912	0
.26179	.965928	.965928	0
.5236	.866025	.866025	0
.75	.731689	.731689	0
.78539	.707113	.707113	0
1	.540302	.540302	0
1.0472	.499998	.499998	0
1.3089	.258913	.258913	0
1.8325	-.258727	-.258727	0
2.0944	-.500004	-.500004	0
2.3561	-.70704	-.70704	0
2.618	-.866028	-.866028	0
2.8797	-.965902	-.965902	0
3	-.989992	-.989992	0
3.1416	-1	-1	0
3.4033	-.96595	-.96595	0
3.6652	-.866021	-.866021	0
3.9279	-.706464	-.706464	0
4.1888	-.499992	-.499992	0
4.4505	-.258906	-.258906	0
4.9741	.258734	.258734	0
5	.283662	.283662	0
5.236	.500011	.500011	0
5.4977	.707045	.707045	0
5.7596	.866032	.866032	0
6.0213	.965904	.965904	0
6.2832	1	1	0
11	4.42570E-3	4.42570E-3	0
25	.991203	.991203	0
75	.921751	.921751	0
1000	.562379	.562377	0
100000	-.999361	-.999361	0

EVALUATION OF THE ABSOLUTE RELATIVE ERRORS

LOWEST ERROR = 0 HIGHEST ERROR = 0

END TEST

98.0 THE EXP FUNCTION

The objective of this test is to introduce the use of the EXP function by sampling the accuracy maintained by the implementation-supplied function (see section 8 of BSR X3.60). EXP computes the value of the base of the natural logarithms raised to the power of the argument. The test, except for the use of the EXP function instead of the SQR function, is the same in structure and output format to section 94.0.

If EXP(X) is less than the local machine infinitesimal, then its value is replaced by zero. The Minimal BASIC standard recommends that underflow be reported as an exception, but it is not required to do so. These remarks are made because the following routine may generate reported underflows on some systems. These occur because of the nonstandard procedures for computing the mathematical functions. A good function should not underflow on this test. The routines should have internal traps that return a function value of 0 when an argument, with the allowed range of numerical representation, produces an underflow.

```
********************
* PROGRAM FILE 98 *
********************
```

```
0010 PRINT "PROGRAM FILE 98"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                              SECTION 98.0"
0100 PRINT
0110 PRINT "                         EXPONENTIAL FUNCTION (EXP)"
0120 PRINT
0130 PRINT "                                   BEGIN TEST"
0140 DATA -.10000E+36 ,  .000000     , -.10000E+11 ,  .000000
0150 DATA -.10000E+03 ,  .000000     , -.80590E+02 ,  .100048E-34
0160 DATA -.50000E+02 ,  .192875E-21 , -.36841E+02 ,  .100036E-15
0170 DATA -.18421E+02 ,  .999681E-08 , -.16000E+02 ,  .112535E-06
0180 DATA -.92103E+01 ,  .100004E-03 , -.80000E+01 ,  .335463E-03
0190 DATA -.46052E+01 ,  .999970E-02 , -.40000E+01 ,  .183156E-01
0200 DATA -.23979E+01 ,  .909087E-01 , -.20000E+01 ,  .135335E+00
0210 DATA -.16094E+01 ,  .200008E+00 , -.10986E+01 ,  .333337E+00
0220 DATA -.33333E+00 ,  .716534E+00 ,  .00000     ,  .100000E+01
0230 DATA  .25000E+00 ,  .128403E+01 ,  .50000E+00 ,  .164872E+01
0240 DATA  .69315E+00 ,  .200001E+01 ,  .10000E+01 ,  .271828E+01
0250 DATA  .13863E+01 ,  .400002E+01 ,  .20794E+01 ,  .799967E+01
0260 DATA  .23026E+01 ,  .100001E+02 ,  .27726E+01 ,  .160002E+02
0270 DATA  .50000E+01 ,  .148413E+03 ,  .69078E+01 ,  .100004E+04
0280 DATA  .11000E+02 ,  .598741E+05 ,  .11513E+02 ,  .100007E+06
0290 DATA  .25000E+02 ,  .720049E+11 ,  .25328E+02 ,  .999564E+11
```

```
0300 DATA  .46052E+02 ,  .100030E+21 ,  .75000E+02 ,  .373324E+33
0310 DATA  .85000E+02 ,  .822301E+37
0320 DIM E(40)
0330 LET L = 0
0340 LET H = 0
0350 LET M = 0
0360 LET V = 0
0370 PRINT " ", " ", " ", "RELATIVE"
0380 PRINT "   ARGUMENT" , " TEST VALUE" , " SYSTEM VALUE" , "  ERROR"
0390 FOR I = 1 TO 35
0395 LET S1 = 1
0400 READ X, Y
0410 LET A$ = " "
0420 LET Z = EXP(X)
0421 IF ABS(Y) >= 1E-38 THEN 428
0422 IF ABS(Z) >= 1E-38 THEN 425
0423 LET E(I) = 0.0
0424 GO TO 500
0425 LET T1 = Y
0426 LET Y = Z
0427 LET Z = T1
0428 REM TEST Y FOR SCALING
0429 IF ABS(Y) <= 1E30 THEN 431
0430 LET S1 = 1E-8
0431 IF ABS(Y) >= 1E-30 THEN 433
0432 LET S1 = 1E8
0433 LET E(I) = ((S1*Z) - (S1*Y))/(S1*Y)
0444 LET E(I) = ABS(E(I))
0445 LET A1 = E(I)
0446 LET Y2 = Y
0447 GOSUB 9000
0448 IF ABS(A1) < C THEN 500
0490 LET A$ = "****"
0500 PRINT X,Y,Z,E(I);A$
0510 NEXT I
0520 PRINT
0530 PRINT "                    EVALUATION OF THE ABSOLUTE RELATIVE";
0540 PRINT " ERRORS"
0550 PRINT
0560 PRINT
0570 FOR I = 1 TO 35
0580 IF H > E(I) THEN 610
0590 LET H = E(I)
0600 GO TO 630
0610 IF L <= E(I) THEN 630
0620 LET L = E(I)
0630 NEXT I
0740 PRINT
0750 PRINT "         LOWEST ERROR =";L;"  HIGHEST ERROR =";H
0760 PRINT
0850 PRINT "                                    END TEST"
0860 PRINT
0880 GO TO 9400
9000 IF 1 <= ABS(Y2) THEN 9101
9001 LET S2 = 1
9003 IF ABS(Y2) > 1E-37 THEN 9010
9005 LET P = 1E-37
```

```
9007 LET S2 = 1E10
9009 GO TO 9200
9010 LET P = 1
9020 LET P1 = P/10
9030 IF P1 <= ABS(Y2) THEN 9200
9040 LET P = P/10
9050 GO TO 9020
9101 LET S2 = 1
9105 IF ABS(Y2) <= 1E37 THEN 9110
9106 LET P = 1E38
9107 LET S2 = 1E-10
9109 GO TO 9200
9110 LET P = 10
9120 IF ABS(Y2) < P THEN 9200
9130 LET P = P*10
9140 GO TO 9120
9200 LET Y1 = ABS(Y2)
9210 LET C = (.5E-6)*((P*S2)/(Y1*S2))
9300 RETURN
9400 END
```

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

PROGRAM FILE 98

SECTION 98.0

EXPONENTIAL FUNCTION (EXP)

BEGIN TEST

ARGUMENT	TEST VALUE	SYSTEM VALUE	RELATIVE ERROR
-1.00000E+35	0	0	0
-1.00000E+10	0	0	0
-100	0	0	0
-80.59	1.00048E-35	1.00048E-35	0
-50	1.92875E-22	1.92875E-22	0
-36.841	1.00036E-16	1.00036E-16	0
-18.421	9.99681E-9	9.99681E-9	0
-16	1.12535E-7	1.12535E-7	0
-9.2103	1.00004E-4	1.00004E-4	0
-8	3.35463E-4	3.35463E-4	0
-4.6052	9.99970E-3	9.99970E-3	0
-4	1.83156E-2	1.83156E-2	0
-2.3979	9.09087E-2	9.09087E-2	0

-2	.135335	.135335	0
-1.6094	.200008	.200008	0
-1.0986	.333337	.333337	0
-.33333	.716534	.716534	0
0	1	1	0
.25	1.28403	1.28403	0
.5	1.64872	1.64872	0
.69315	2.00001	2.00001	0
1	2.71828	2.71828	0
1.3863	4.00002	4.00002	0
2.0794	7.99967	7.99967	0
2.3026	10.0001	10.0001	0
2.7726	16.0002	16.0002	0
5	148.413	148.413	0
6.9078	1000.04	1000.04	0
11	59874.1	59874.1	0
11.513	100007	100007	0
25	7.20049E+10	7.20049E+10	0
25.328	9.99564E+10	9.99564E+10	0
46.052	1.00030E+20	1.00030E+20	0
75	3.73324E+32	3.73324E+32	0
85	8.22301E+36	8.22301E+36	0

EVALUATION OF THE ABSOLUTE RELATIVE ERRORS

LOWEST ERROR = 0 HIGHEST ERROR = 0

END TEST

99.0 EXCEPTION TEST FOR THE EXP FUNCTION

The objective of this test is to determine whether the implementation recognizes certain exceptions for the EXP function (see section 8.5 of BSR X3.60). In particular, this section examines two cases when the evaluation of the Exponential Function results in an overflow. If the implementation recognizes, reports and assigns the proper sign to its implementation-defined machine infinity for any overflow, then the system will have passed the test.

99.1 Positive Machine-Infinity

This test uses an argument value of 99999 for the EXP function in line 380. For most machines available, this value should cause an overflow. On output, the test prints a message warning the user to look for a printout of the positive case of the implementation-defined machine-infinity for the system being tested, since an unsigned EXP function is used in line 380. This use of an EXP function means that a positive machine infinity must be returned. There should also be some diagnostic report indicating that machine overflow was encountered.

99.2 Negative Machine-Infinity

This test has a negative assignment of the EXP function, using an argument value of 99999 in line 520. On output, the test has a message which asks the user to look for a printout of the negative case of the implementation-defined machine infinity for the system being tested. A diagnostic indicating machine overflow should also be printed.

```
********************
* PROGRAM FILE 99 *
********************
```

```
0010 PRINT "PROGRAM FILE 99"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "  SECTION 99.0: EXCEPTION TEST FOR THE EXP FUNCTION."
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0140 PRINT "THIS SYSTEM RECOGNIZES THE FOLLOWING PROCEDURE:"
0160 PRINT "     TO SUPPLY MACHINE INFINITY WITH THE APPROPRIATE SIGN"
0170 PRINT "AND CONTINUE PROGRAM EXECUTION IF THE EVALUATION OF THE EX-"
0180 PRINT "PONENTIAL FUNCTION RESULTS IN AN OVERFLOW."
```

```
0220 PRINT
0230 PRINT "IF SUCH RECOGNITION IS MADE BY THIS SYSTEM, THEN CONSIDER"
0240 PRINT "IT TO HAVE PASSED THE TEST."
0250 PRINT
0260 PRINT
0270 PRINT
0340 PRINT "              SECTION 99.1: POSITIVE MACHINE INFINITY."
0350 PRINT
0360 PRINT "                                   BEGIN TEST."
0370 PRINT
0380 LET Y=EXP(99999)
0390 PRINT "IF THE NUMERICAL VALUE FOR THE POSITIVE MACHINE INFINITY"
0400 PRINT "OF THIS SYSTEM FOLLOWS THIS STATEMENT, THEN CONSIDER THE"
0410 PRINT "SYSTEM TO HAVE PASSED THE TEST."
0420 PRINT Y
0430 PRINT
0440 PRINT "                                   END TEST."
0450 PRINT
0460 PRINT
0470 PRINT
0480 PRINT "              SECTION 99.2: NEGATIVE MACHINE INFINITY."
0490 PRINT
0500 PRINT "                                   BEGIN TEST."
0510 PRINT
0520 LET Y=-EXP(99999)
0530 PRINT "IF THE NUMERICAL VALUE FOR THE NEGATIVE MACHINE INFINITY"
0540 PRINT "OF THIS SYSTEM FOLLOWS THIS STATEMENT, THEN CONSIDER THE"
0550 PRINT "SYSTEM TO HAVE PASSED THE TEST."
0560 PRINT Y
0570 PRINT
0580 END
```

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

```
PROGRAM FILE 99


  SECTION 99.0: EXCEPTION TEST FOR THE EXP FUNCTION.


     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER
THIS SYSTEM RECOGNIZES THE FOLLOWING PROCEDURE:
     TO SUPPLY MACHINE INFINITY WITH THE APPROPRIATE SIGN
AND CONTINUE PROGRAM EXECUTION IF THE EVALUATION OF THE EX-
PONENTIAL FUNCTION RESULTS IN AN OVER FLOW.
```

IF SUCH RECOGNITION IS MADE BY THIS SYSTEM, THEN CONSIDER
IT TO HAVE PASSED THE TEST.

SECTION 99.1: POSITIVE MACHINE INFINITY.

BEGIN TEST.

?OVERFLOW IN EXP IN LINE 380
IF THE NUMERICAL VALUE FOR THE POSITIVE MACHINE INFINITY
OF THIS SYSTEM FOLLOWS THIS STATEMENT, THEN CONSIDER THE
SYSTEM TO HAVE PASSED THE TEST.
 1.70141E+38

END TEST.

SECTION 99.2: NEGATIVE MACHINE INFINITY.

BEGIN TEST.

?OVERFLOW IN EXP IN LINE 520
IF THE NUMERICAL VALUE FOR THE NEGATIVE MACHINE INFINITY
OF THIS SYSTEM FOLLOWS THIS STATEMENT, THEN CONSIDER THE
SYSTEM TO HAVE PASSED THE TEST.
-1.70141E+38

100.0 UNDERFLOW OF THE EXPONENTIAL FUNCTION

This test verifies that the evaluation of an EXP function causing underflow will be recognized, and a value assignment of zero will be given by the implementation to the EXP function (see section 8.6 of BSR X3.60). -99999 in line 170 is assigned as an argument value for the EXP function. On output, the test has a message which warns the user to look for a printout value of zero by the system being tested. The standard recommends that a system report an underflow when the exponential function produces a machine infinitessimal value or less.

****
* PROGRAM FILE 100 *
****

```
0010 PRINT "PROGRAM FILE 100"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 100.0"
0100 PRINT
0110 PRINT "(EVALUATION OF THE EXPONENTIAL FUNCTION RESULTS IN AN UNDER"
0120 PRINT " FLOW.)"
0130 PRINT
0140 PRINT
0150 PRINT "                                BEGIN TEST."
0160 PRINT
0170 LET Y=EXP(-999999)
0180 PRINT "IF THE NUMERICAL VALUE OF ZERO FOLLOWS THIS STATEMENT, THEN"
0190 PRINT "CONSIDER THE SYSTEM TO HAVE PASSED TEST."
0200 PRINT Y
0210 PRINT
0220 PRINT "                                 END TEST."
0230 PRINT
0240 END
```

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

PROGRAM FILE 100

SECTION 100.0

(EVALUATION OF THE EXPONENTIAL FUNCTION RESULTS IN AN UNDER
FLOW.)

BEGIN TEST.

?UNDERFLOW IN EXP IN LINE 170
IF THE NUMERICAL VALUE OF ZERO FOLLOWS THIS STATEMENT, THEN
CONSIDER THE SYSTEM TO HAVE PASSED TEST.
0

END TEST.

101.0 THE LOG FUNCTION

This test introduces the use of the LOG function by sampling the accuracy of this implementation-supplied function. The function returns the natural logarithm of an argument that must be greater than zero. The test, except for the use of the LOG function instead of the SQR function, is similar in structure and output format to section 94.0. The reader is referred to section 8 of BSR X3.60.

****
* PROGRAM FILE 101 *
****

```
0010 PRINT "PROGRAM FILE 101"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                              SECTION 101.0"
0100 PRINT
0110 PRINT "                          NATURAL LOG FUNCTION (LOG)"
0120 PRINT
0130 PRINT "                                BEGIN TEST"
0140 DATA  .10000E-19 , -.460517E+02 ,  .10000E-07 , -.184207E+02
0150 DATA  .11254E-06 , -.160000E+02 ,  .10000E-04 , -.115129E+02
0160 DATA  .33546E-03 , -.800001E+01 ,  .10000E-01 , -.460517E+01
0170 DATA  .18316E-01 , -.399998E+01 ,  .31250E-01 , -.346574E+01
0180 DATA  .90909E-01 , -.239790E+01 ,  .13534E+00 , -.199997E+01
0190 DATA  .20000E+00 , -.160944E+01 ,  .33333E+00 , -.109862E+01
0200 DATA  .50000E+00 , -.693147E+00 ,  .75000E+00 , -.287682E+00
0210 DATA  .90000E+00 , -.105361E+00 ,  .99000E+00 , -.100503E-01
0220 DATA  .99900E+00 , -.100050E-02 ,  .99990E+00 , -.100005E-03
0230 DATA  .99999E+00 , -.100001E-04 ,  .10000E+01 ,  .000000
0240 DATA  .10001E+01 ,  .999950E-04 ,  .10010E+01 ,  .999500E-03
0250 DATA  .10100E+01 ,  .995033E-02 ,  .11000E+01 ,  .953102E-01
0260 DATA  .15000E+01 ,  .405465E+00 ,  .20000E+01 ,  .693147E+00
0270 DATA  .27183E+01 ,  .100001E+01 ,  .40000E+01 ,  .138629E+01
0280 DATA  .80000E+01 ,  .207944E+01 ,  .10000E+02 ,  .230259E+01
0290 DATA  .16000E+02 ,  .277259E+01 ,  .20086E+02 ,  .300002E+01
0300 DATA  .64000E+02 ,  .415888E+01 ,  .14841E+03 ,  .499998E+01
0310 DATA  .10000E+04 ,  .690776E+01 ,  .10000E+05 ,  .921034E+01
0320 DATA  .59874E+05 ,  .110000E+02 ,  .10000E+11 ,  .230259E+02
0330 DATA  .10000E+36 ,  .805905E+02
0340 DIM E(39)
0350 LET L = 0
0360 LET H = 0
0370 LET M = 0
0380 LET V = 0
0390 PRINT " ", " ", " ", "RELATIVE"
```

```
0400 PRINT "   ARGUMENT", " TEST VALUE", " SYSTEM VALUE", "  ERROR"
0410 FOR I = 1 TO 39
0415 LET S1 = 1
0420 READ X,Y
0430 LET A$ = " "
0440 LET Z = LOG(X)
0441 IF ABS(Y) >= 1E-38 THEN 448
0442 IF ABS(Z) >= 1E-38 THEN 445
0443 LET E(I) = 0.0
0444 GO TO 520
0445 LET T1 = Y
0446 LET Y = Z
0447 LET Z = T1
0448 REM TEST Y FOR SCALING
0449 IF ABS(Y) <= 1E30 THEN 451
0450 LET S1 = 1E-8
0451 IF ABS(Y) >= 1E-30 THEN 453
0452 LET S1 = 1E8
0453 LET E(I) = ((S1*Z) - (S1*Y))/(S1*Y)
0454 LET E(I) = ABS(E(I))
0455 LET A1 = E(I)
0456 LET Y2 = Y
0457 GOSUB 9000
0458 IF ABS(A1) < C THEN 520
0510 LET A$ = "****"
0520 PRINT X,Y,Z,E(I);A$
0530 NEXT I
0540 PRINT
0550 PRINT "               EVALUATION OF THE ABSOLUTE RELATIVE";
0560 PRINT " ERRORS"
0570 PRINT
0580 PRINT
0590 FOR I = 1 TO 39
0600 IF H > E(I) THEN 630
0610 LET H = E(I)
0620 GO TO 650
0630 IF L <= E(I) THEN 650
0640 LET L = E(I)
0650 NEXT I
0760 PRINT
0770 PRINT "        LOWEST ERROR = ";L;"  HIGHEST ERROR =";H
0780 PRINT
0870 PRINT "                                   END TEST"
0880 PRINT
0890 GO TO 9400
9000 IF 1 <= ABS(Y2) THEN 9101
9001 LET S2 = 1
9003 IF ABS(Y2) > 1E-37 THEN 9010
9005 LET P = 1E-37
9007 LET S2 = 1E10
9009 GO TO 9200
9010 LET P = 1
9020 LET P1 = P/10
9030 IF P1 <= ABS(Y2) THEN 9200
9040 LET P = P/10
9050 GO TO 9020
9101 LET S2 = 1
```

```
9105 IF ABS(Y2) <= 1E37 THEN 9110
9106 LET P = 1E38
9107 LET S2 = 1E-10
9109 GO TO 9200
9110 LET P = 10
9120 IF ABS(Y2) < P THEN 9200
9130 LET P = P*10
9140 GO TO 9120
9200 LET Y1 = ABS(Y2)
9210 LET C = (.5E-6)*((P*S2)/(Y1*S2))
9300 RETURN
9400 END
```

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

PROGRAM FILE 101

SECTION 101.0

NATURAL LOG FUNCTION (LOG)

BEGIN TEST

ARGUMENT	TEST VALUE	SYSTEM VALUE	RELATIVE ERROR
1.00000E-20	-46.0517	-46.0517	0
1.00000E-8	-18.4207	-18.4207	0
1.12540E-7	-16	-16	0
.00001	-11.5129	-11.5129	0
3.35460E-4	-8.00001	-8.00001	0
.01	-4.60517	-4.60517	0
.018316	-3.99998	-3.99998	0
3.12500E-2	-3.46574	-3.46574	0
9.09090E-2	-2.3979	-2.3979	0
.13534	-1.99997	-1.99997	0
.2	-1.60944	-1.60944	0
.33333	-1.09862	-1.09862	0
.5	-.693147	-.693147	0
.75	-.287682	-.287682	0
.9	-.105361	-.105361	0
.99	-1.00503E-2	-1.00503E-2	0
.999	-1.00050E-3	-1.00050E-3	0
.9999	-1.00005E-4	-1.00005E-4	0
.99999	-1.00001E-5	-1.00001E-5	0
1	0	0	0
1.0001	9.99950E-5	9.99950E-5	0

1.001	9.99500E-4	9.99500E-4	0
1.01	9.95033E-3	9.95033E-3	0
1.1	9.53102E-2	9.53102E-2	0
1.5	.405465	.405465	0
2	.693147	.693147	0
2.7183	1.00001	1.00001	0
4	1.38629	1.38629	0
8	2.07944	2.07944	0
10	2.30259	2.30259	0
16	2.77259	2.77259	0
20.086	3.00002	3.00002	0
64	4.15888	4.15888	0
148.41	4.99998	4.99998	0
1000	6.90776	6.90776	0
10000	9.21034	9.21034	0
59874	11	11	0
1.00000E+10	23.0259	23.0259	0
1.00000E+35	80.5905	80.5905	0

EVALUATION OF THE ABSOLUTE RELATIVE ERRORS

LOWEST ERROR = 0 HIGHEST ERROR = 0

END TEST

102.0 EXCEPTION TEST FOR THE LOG FUNCTION WITH A ZERO ARGUMENT

The objective of this test is to verify that the evaluation of the LOG function with a zero argument will be recognized as an exception by the implementation being tested (see section 8.5 of BSR X3.60). This exception when encountered must be reported by a Minimal BASIC processor. To test this requirement the program below assigns a LOG function with an argument value of zero in line 290. According to the ANSI Minimal BASIC standard, this exception should be handled by the implementation first by reporting the exception, and then terminating the program.

****
* PROGRAM FILE 102 *
****

```
0010 PRINT "PROGRAM FILE 102"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "     SECTION 102.0: EXCEPTION TEST FOR THE LOG FUNCTION"
0100 PRINT "                    WHEN THE ARGUMENT IS ZERO."
0110 PRINT
0120 PRINT
0130 PRINT "     THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER"
0140 PRINT "           THIS SYSTEM RECOGNIZES THE USE OF A ZERO  "
0150 PRINT "     FOR THE ARGUMENT OF THE LOG FUNCTION AS AN EXCEPTION"
0160 PRINT "THE SYSTEM MUST REPORT THE EXCEPTION AND TERMINATE PROGRAM"
0190 PRINT
0200 PRINT
0210 PRINT
0270 PRINT "                                BEGIN TEST."
0280 PRINT
0290 LET Y=LOG(0)
0300 PRINT "IF A NUMERICAL VALUE IS PRINTED AFTER THIS STATEMENT,"
0310 PRINT "CONSIDER SYSTEM TO HAVE FAILED TEST."
0320 PRINT Y
0330 PRINT
0340 PRINT "                                 END TEST."
0350 PRINT
0360 END
```

****
* SAMPLE OUTPUT *

****

PROGRAM FILE 102

SECTION 102.0: EXCEPTION TEST FOR THE LOG FUNCTION
WHEN THE ARGUMENT IS ZERO.

THE OBJECTIVE OF THIS SECTION IS TO DETERMINE WHETHER
THIS SYSTEM RECOGNIZES THE USE OF A ZERO
FOR THE ARGUMENT OF THE LOG FUNCTION AS AN EXCEPTION
THE SYSTEM MUST REPORT THE EXCEPTION AND TERMINATE PROGRAM

BEGIN TEST.

?LOG OF ZERO IN LINE 290

103.0 EXCEPTION TEST FOR THE LOG FUNCTION WITH A NEGATIVE ARGUMENT

The objective of this test is to verify that the implementation will recognize as an exception the evaluation of the LOG function with a negative value as an argument (see section 8.5 of BSR X3.60).

This test has an assignment of LOG using a negative argument in line 160. Some present systems, however, ignore the negative value and return the LOG of the absolute value of the argument. This would be considered a failure, since the program should have terminated with a report on the exception. In fact, on output, there should be implementation-defined diagnostic. Otherwise, the test program below has an internal message indicating system failure.

****
* PROGRAM FILE 103 *
****

```
0010 PRINT "PROGRAM FILE 103"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 103.0"
0100 PRINT "                  EXCEPTION TEST FOR THE LOG FUNCTION"
0110 PRINT "                 THE VALUE OF THE ARGUMENT IS NEGATIVE. "
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 LET Y=LOG(-16)
0170 PRINT "IF A NUMBERICAL VALUE IS PRINTED AFTER THIS STATEMENT,"
0180 PRINT "CONSIDER SYSTEM TO HAVE FAILED TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                   END TEST."
0220 PRINT
0230 END
```

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

PROGRAM FILE 103

SECTION 103.0
EXCEPTION TEST FOR THE LOG FUNCTION
THE VALUE OF THE ARGUMENT IS NEGATIVE.

BEGIN TEST.

?LOG OF NEGATIVE NUMBER IN LINE 160

104.0 THE SIN FUNCTION

The object of this test is to introduce the use of the SIN function with a test sampling the accuracy maintained by the implementation-supplied function (see section 8 of BSR X3.60). Every allowable numeric value is a legitimate argument. Arguments are of course assumed to be radians. The actual test is similar to section 94.0.

****
* PROGRAM FILE 104 *
****

```
0010 PRINT "PROGRAM FILE 104"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                              SECTION 104.0"
0100 PRINT
0110 PRINT "                            SINE FUNCTION (SIN)"
0120 PRINT
0130 PRINT "                                  BEGIN TEST"
0140 DATA -.10000E+05 ,  .305614E+00 , -.10000E+03 ,  .506366E+00
0150 DATA -.50000E+02 ,  .262375E+00 , -.16000E+02 ,  .287903E+00
0160 DATA -.80000E+01 , -.989358E+00 , -.40000E+01 ,  .756803E+00
0170 DATA -.20000E+01 , -.909297E+00 , -.50000E+00 , -.479426E+00
0180 DATA -.10000E+00 , -.998334E-01 , -.10000E-02 , -.100000E-02
0190 DATA -.10000E-04 , -.100000E-04 , -.10000E-10 , -.100000E-10
0200 DATA -.10000E-24 , -.100000E-24 ,  .00000     ,  .000000
0210 DATA  .10000E-34 ,  .100000E-34 ,  .10000E-15 ,  .100000E-15
0220 DATA  .10000E-07 ,  .100000E-07 ,  .10000E-03 ,  .100000E-03
0230 DATA  .10000E-01 ,  .999983E-02 ,  .25000E+00 ,  .247404E+00
0240 DATA  .26179E+00 ,  .258810E+00 ,  .52360E+00 ,  .500001E+00
0250 DATA  .75000E+00 ,  .681639E+00 ,  .78539E+00 ,  .707101E+00
0260 DATA  .10000E+01 ,  .841471E+00 ,  .10472E+01 ,  .866027E+00
0270 DATA  .13089E+01 ,  .965901E+00 ,  .15708E+01 ,  .100000E+01
0280 DATA  .18325E+01 ,  .965951E+00 ,  .20944E+01 ,  .866023E+00
0290 DATA  .23561E+01 ,  .707174E+00 ,  .26180E+01 ,  .499995E+00
0300 DATA  .28797E+01 ,  .258909E+00 ,  .30000E+01 ,  .141120E+00
0310 DATA  .34033E+01 , -.258730E+00 ,  .36652E+01 , -.500007E+00
0320 DATA  .39279E+01 , -.707749E+00 ,  .41888E+01 , -.866030E+00
0330 DATA  .44505E+01 , -.965903E+00 ,  .47124E+01 , -.100000E+01
0340 DATA  .49741E+01 , -.965949E+00 ,  .50000E+01 , -.958924E+00
0350 DATA  .52360E+01 , -.866019E+00 ,  .54977E+01 , -.707168E+00
0360 DATA  .57596E+01 , -.499988E+00 ,  .60213E+01 , -.258902E+00
0370 DATA  .11000E+02 , -.999990E+00 ,  .25000E+02 , -.132352E+00
0380 DATA  .75000E+02 , -.387782E+00 ,  .10000E+04 ,  .826880E+00
0390 DATA  .10000E+06 ,  .357488E-01
0400 DIM E(51)
```

```
0410 LET L = 0
0420 LET H = 0
0430 LET M = 0
0440 LET V = 0
0450 PRINT " ", " ", " ", "RELATIVE"
0460 PRINT "   ARGUMENT", " TEST VALUE", " SYSTEM VALUE", "  ERROR"
0470 FOR I = 1 TO 51
0475 LET S1 = 1
0480 READ X,Y
0490 LET A$ = " "
0500 LET Z = SIN(X)
0501 IF ABS(Y) >= 1E-38 THEN 508
0502 IF ABS(Z) >= 1E-38 THEN 505
0503 LET E(I) = 0.0
0504 GO TO 580
0505 LET T1 = Y
0506 LET Y = Z
0507 LET Z = T1
0508 REM TEST Y FOR SCALING
0509 IF ABS(Y) <= 1E30 THEN 511
0510 LET S1 = 1E-8
0511 IF ABS(Y) >= 1E-30 THEN 513
0512 LET S1 = 1E8
0513 LET E(I) = ((S1*Z) - (S1*Y))/(S1*Y)
0514 LET E(I) = ABS(E(I))
0515 LET A1 = E(I)
0516 LET Y2 = Y
0517 GOSUB 9000
0518 IF ABS(A1) < C THEN 580
0570 LET A$ = "****"
0580 PRINT X,Y,Z,E(I);A$
0590 NEXT I
0600 PRINT
0610 PRINT "          EVALUATION OF THE ABSOLUTE RELATIVE";
0620 PRINT " ERRORS"
0630 PRINT
0640 PRINT
0650 FOR I = 1 TO 51
0660 IF H > E(I) THEN 690
0670 LET H = E(I)
0680 GO TO 710
0690 IF L <= E(I) THEN 710
0700 LET L = E(I)
0710 NEXT I
0820 PRINT
0830 PRINT "         LOWEST ERROR = ";L;" HIGHEST ERROR = ";H
0840 PRINT
0930 PRINT "                                   END TEST"
0940 PRINT
0950 GO TO 9400
9000 IF 1 <= ABS(Y2) THEN 9101
9001 LET S2 = 1
9003 IF ABS(Y2) > 1E-37 THEN 9010
9005 LET P = 1E-37
9007 LET S2 = 1E10
9009 GO TO 9200
9010 LET P = 1
```

```
9020 LET P1 = P/10
9030 IF P1 <= ABS(Y2) THEN 9200
9040 LET P = P/10
9050 GO TO 9020
9101 LET S2 = 1
9105 IF ABS(Y2) <= 1E37 THEN 9110
9106 LET P = 1E38
9107 LET S2 = 1E-10
9109 GO TO 9200
9110 LET P = 10
9120 IF ABS(Y2) < P THEN 9200
9130 LET P = P*10
9140 GO TO 9120
9200 LET Y1 = ABS(Y2)
9210 LET C = (.5E-6)*((P*S2)/(Y1*S2))
9300 RETURN
9400 END
```

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

PROGRAM FILE 104

SECTION 104.0

SINE FUNCTION (SIN)

BEGIN TEST

ARGUMENT	TEST VALUE	SYSTEM VALUE	RELATIVE ERROR
-10000	.305614	.305614	0
-100	.506366	.506366	0
-50	.262375	.262375	0
-16	.287903	.287903	0
-8	-.989358	-.989358	0
-4	.756803	.756803	0
-2	-.909297	-.909297	0
-.5	-.479426	-.479426	0
-.1	-9.98334E-2	-9.98334E-2	0
-.001	-.001	-.001	0
-.00001	-.00001	-.00001	0
-1.00000E-11	-1.00000E-11	-1.00000E-11	0
-1.00000E-25	-1.00000E-25	-1.00000E-25	0
0	0	0	0
1.00000E-35	1.00000E-35	1.00000E-35	0
1.00000E-16	1.00000E-16	1.00000E-16	0

1.00000E-8	1.00000E-8	1.00000E-8	0
.0001	.0001	.0001	0
.01	9.99983E-3	9.99983E-3	0
.25	.247404	.247404	0
.26179	.25881	.25881	0
.5236	.500001	.500001	0
.75	.681639	.681639	0
.78539	.707101	.707101	0
1	.841471	.841471	0
1.0472	.866027	.866027	0
1.3089	.965901	.965901	0
1.5708	1	1	0
1.8325	.965951	.965951	0
2.0944	.866023	.866023	0
2.3561	.707174	.707174	0
2.618	.499995	.499995	0
2.8797	.258909	.258909	0
3	.14112	.14112	0
3.4033	-.25873	-.25873	0
3.6652	-.500007	-.500007	0
3.9279	-.707749	-.707749	0
4.1888	-.86603	-.86603	0
4.4505	-.965903	-.965903	0
4.7124	-1	-1	0
4.9741	-.965949	-.965949	0
5	-.958924	-.958924	0
5.236	-.866019	-.866019	0
5.4977	-.707168	-.707168	0
5.7596	-.499988	-.499988	0
6.0213	-.258902	-.258902	0
11	-.99999	-.99999	0
25	-.132352	-.132352	0
75	-.387782	-.387782	0
1000	.82688	.82688	0
100000	3.57488E-2	3.57488E-2	0

EVALUATION OF THE ABSOLUTE RELATIVE ERRORS

LOWEST ERROR = 0 HIGHEST ERROR = 0

END TEST

105.0 THE TAN FUNCTION

This test introduce the use of tne TAN function by sampling the accuracy maintained by the implementation-supplied function (see section 8 of BSR X3.60). The argument in the tangent function is assumed to be in radians. Since the tangent function becomes unbounbed near odd multipes of pi/2, it is at these points in the testing that the user should look for accuracy deterioration. There might also be loss of accuracy near the zeroes of the tangent function too. The actual test is similar to section 94.0.

```
**********************
* PROGRAM FILE 105 *
**********************
```

```
0010 PRINT "PROGRAM FILE 105"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                              SECTION 105.0"
0100 PRINT
0110 PRINT "                          TANGENT FUNCTION (TAN)"
0120 PRINT
0130 PRINT "                                 BEGIN TEST"
0140 DATA -.10000E+05 , -.320971E+00 , -.10000E+03 ,  .587214E+00
0150 DATA -.50000E+02 ,  .271901E+00 , -.16000E+02 , -.300632E+00
0160 DATA -.80000E+01 ,  .679971E+01 , -.40000E+01 , -.115782E+01
0170 DATA -.20000E+01 ,  .218504E+01 , -.50000E+00 , -.546302E+00
0180 DATA -.10000E+00 , -.100335E+00 , -.10000E-02 , -.100000E-02
0190 DATA -.10000E-04 , -.100000E-04 , -.10000E-10 , -.100000E-10
0200 DATA -.10000E-24 , -.100000E-24 ,  .00000     ,  .000000
0210 DATA  .10000E-34 ,  .100000E-34 ,  .10000E-15 ,  .100000E-15
0220 DATA  .10000E-07 ,  .100000E-07 ,  .10000E-03 ,  .100000E-03
0230 DATA  .10000E-01 ,  .100003E-01 ,  .25000E+00 ,  .255342E+00
0240 DATA  .26179E+00 ,  .267939E+00 ,  .52360E+00 ,  .577352E+00
0250 DATA  .75000E+00 ,  .931596E+00 ,  .78539E+00 ,  .999984E+00
0260 DATA  .10000E+01 ,  .155741E+01 ,  .10472E+01 ,  .173206E+01
0270 DATA  .13089E+01 ,  .373060E+01 ,  .15708E+01 , -.272242E+06
0280 DATA  .18325E+01 , -.373348E+01 ,  .20944E+01 , -.173203E+01
0290 DATA  .23561E+01 , -.100019E+01 ,  .26180E+01 , -.577342E+00
0300 DATA  .28797E+01 , -.268049E+00 ,  .30000E+01 , -.142547E+00
0310 DATA  .31416E+01 ,  .734641E-05 ,  .34033E+01 ,  .267851E+00
0320 DATA  .36652E+01 ,  .577362E+00 ,  .39279E+01 ,  .100182E+01
0330 DATA  .41888E+01 ,  .173209E+01 ,  .44505E+01 ,  .373071E+01
0340 DATA  .47124E+01 , -.907473E+05 ,  .49741E+01 , -.373337E+01
0350 DATA  .50000E+01 , -.338052E+01 ,  .52360E+01 , -.173200E+01
0360 DATA  .54977E+01 , -.100017E+01 ,  .57596E+01 , -.577352E+00
0370 DATA  .60213E+01 , -.268041E+00 ,  .62832E+01 ,  .146928E-04
0380 DATA  .11000E+02 , -.225951E+03 ,  .25000E+02 , -.133526E+00
```

```
0390 DATA  .75000E+02 , -.420701E+00 ,  .10000E+04 ,  .147032E+01
0400 DATA  .10000E+06 , -.357717E-01
0410 DIM E(53)
0420 LET L = 0
0430 LET H = 0
0440 LET M = 0
0450 LET V = 0
0460 PRINT " ", " ", " ", "RELATIVE"
0470 PRINT "   ARGUMENT", "TEST VALUE", " SYSTEM VALUE", "  ERROR"
0480 FOR I = 1 TO 53
0485 LET S1 = 1
0490 READ X,Y
0500 LET A$ = " "
0510 LET Z = TAN(X)
0511 IF ABS(Y) >= 1E-38 THEN 518
0512 IF ABS(Z) >= 1E-38 THEN 515
0513 LET E(I) = 0.0
0514 GO TO 590
0515 LET T1 = Y
0516 LET Y = Z
0517 LET Z = T1
0518 REM TEST Y FOR SCALING
0519 IF ABS(Y) <= 1E30 THEN 521
0520 LET S1 = 1E-8
0521 IF ABS(Y) >= 1E-30 THEN 523
0522 LET S1 = 1E8
0523 LET E(I) = ((S1*Z) - (S1*Y))/(S1*Y)
0524 LET E(I) = ABS(E(I))
0525 LET A1 = E(I)
0526 LET Y2 = Y
0527 GOSUB 9000
0528 IF ABS(A1) < C THEN 590
0580 LET A$ = "****"
0590 PRINT X,Y,Z,E(I);A$
0600 NEXT I
0610 PRINT
0620 PRINT "              EVALUATION OF THE ABSOLUTE RELATIVE";
0630 PRINT " ERRORS"
0640 PRINT
0650 PRINT
0660 FOR I = 1 TO 53
0670 IF H > E(I) THEN 700
0680 LET H = E(I)
0690 GO TO 720
0700 IF L <= E(I) THEN 720
0710 LET L = E(I)
0720 NEXT I
0830 PRINT
0840 PRINT "        LOWEST ERROR = ";L;" HIGHEST ERROR = ";H
0850 PRINT
0940 PRINT "                                    END TEST"
0950 PRINT
0960 GO TO 9400
9000 IF 1 <= ABS(Y2) THEN 9101
9001 LET S2 = 1
9003 IF ABS(Y2) > 1E-37 THEN 9010
9005 LET P = 1E-37
```

```
9007 LET S2 = 1E10
9009 GO TO 9200
9010 LET P = 1
9020 LET P1 = P/10
9030 IF P1 <= ABS(Y2) THEN 9200
9040 LET P = P/10
9050 GO TO 9020
9101 LET S2 = 1
9105 IF ABS(Y2) <= 1E37 THEN 9110
9106 LET P = 1E38
9107 LET S2 = 1E-10
9109 GO TO 9200
9110 LET P = 10
9120 IF ABS(Y2) < P THEN 9200
9130 LET P = P*10
9140 GO TO 9120
9200 LET Y1 = ABS(Y2)
9210 LET C = (.5E-6)*((P*S2)/(Y1*S2))
9300 RETURN
9400 END
```

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

PROGRAM FILE 105

SECTION 105.0

TANGENT FUNCTION (TAN)

BEGIN TEST

ARGUMENT	TEST VALUE	SYSTEM VALUE	RELATIVE ERROR
-10000	-.320971	-.320971	0
-100	.587214	.587214	0
-50	.271901	.271901	0
-16	-.300632	-.300632	0
-8	6.79971	6.79971	0
-4	-1.15782	-1.15782	0
-2	2.18504	2.18504	0
-.5	-.546302	-.546302	0
-.1	-.100335	-.100335	0
-.001	-.001	-.001	0
-.00001	-.00001	-.00001	0
-1.00000E-11	-1.00000E-11	-1.00000E-11	0
-1.00000E-25	-1.00000E-25	-1.00000E-25	0

0	0	0	0
1.00000E-35	1.00000E-35	1.00000E-35	0
1.00000E-16	1.00000E-16	1.00000E-16	0
1.00000E-8	1.00000E-8	1.00000E-8	0
.0001	.0001	.0001	0
.01	1.00003E-2	1.00003E-2	0
.25	.255342	.255342	0
.26179	.267939	.267939	0
.5236	.577352	.577352	0
.75	.931596	.931596	0
.78539	.999984	.999984	0
1	1.55741	1.55741	0
1.0472	1.73206	1.73206	0
1.3089	3.7306	3.7306	0
1.5708	-272242	-272242	0
1.8325	-3.73348	-3.73348	0
2.0944	-1.73203	-1.73203	0
2.3561	-1.00019	-1.00019	0
2.618	-.577342	-.577342	0
2.8797	-.268049	-.268049	0
3	-.142547	-.142547	0
3.1416	7.34641E-6	7.34641E-6	0
3.4033	.267851	.267851	0
3.6652	.577362	.577362	0
3.9279	1.00182	1.00182	0
4.1888	1.73209	1.73209	0
4.4505	3.73071	3.73071	0
4.7124	-90747.3	-90747.3	0
4.9741	-3.73337	-3.73337	0
5	-3.38052	-3.38052	0
5.236	-1.732	-1.732	0
5.4977	-1.00017	-1.00017	0
5.7596	-.577332	-.577332	0
6.0213	-.268041	-.268041	0
6.2832	1.46928E-5	1.46928E-5	0
11	-225.951	-225.951	0
25	-.133526	-.133526	0
75	-.420701	-.420701	0
1000	1.47032	1.47032	0
100000	-3.57717E-2	-3.57717E-2	0

EVALUATION OF THE ABSOLUTE RELATIVE ERRORS

LOWEST ERROR = 0 HIGHEST ERROR = 0

END TEST

106.0 ACCURACY FOR EXPONENTIATION

Exponentiation has been used previously to test the operations and their hierarchy. This test is aimed at assessing some of the accuracy considerations relating to exponentiation. We have sampled 17 pairs of X, Y values covering some of the range of allowed values, and then we compute X^Y to compare this against a test value. The program gives five columns of output. The items generated are: the X argument, the Y exponent, the test value, and the system generated-value. The last column gives the relative error between the test and system value. If the absolute relative error is out of tolerance, then asterisks will appear next to the relative error, indicating that there is loss of significance in the system result. Accuracy beyond 6 significant digits is not tested in this program.

```
**********************
* PROGRAM FILE 106 *
**********************
```

```
10 PRINT "PROGRAM FILE 106"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "                                SECTION 106.0"
60 PRINT
70 PRINT "                                EXPONENTIATION"
80 PRINT "                                     X^Y"
90 PRINT
100 PRINT "                                   BEGIN TEST"
110 DATA 1.0E38, 0.5, 1.0E19
120 DATA 1.0E38, -1, 1.0E-38
130 DATA 9.99999E18, 2, 9.99998E37
140 DATA 100, 7, 1.0E14
150 DATA 100, -7, 1.0E-14
160 DATA 10, 24, 1.0E24
170 DATA 10, -24, 1.0E-24
180 DATA 2, 89, 6.1897E26
190 DATA 2, 66.6, 1.1184E20
200 DATA 2, 16, 65536
210 DATA 2, -44.3, 4.61712E-14
220 DATA 1.0001, 100, 1.01005
230 DATA 1, 88, 1
240 DATA 1, -1, 1
250 DATA 0.9999, 77.777, 0.992252
260 DATA 1.73715E-11, 3, 5.24218E-33
270 DATA 1.0E-38, 0.5, 1.0E-19
280 DIM E(17)
290 LET L = 0
300 LET H = 0
```

```
310 PRINT " ", " ", " ", " ", "RELATIVE"
320 PRINT "X", "Y", "TEST VALUE", "SYSTEM VALUE", "ERROR"
330 FOR I = 1 TO 17
335 LET S1 = 1
340 READ X, Y, T
350 LET A$ = " "
360 LET Z = X^Y
370 IF T >= 1E-38 THEN 395
375 IF Z >= 1E-38 THEN 392
380 LET E(I) = 0.0
390 GO TO 450
392 LET T1 = T
393 LET T = Z
394 LET Z = T1
395 REM TEST T FOR SCALING
396 IF ABS(T) <= 1E30 THEN 398
397 LET S1 = 1E-8
398 IF ABS(T) >= 1E-30 THEN 400
399 LET S1 = 1E8
400 LET E(I) = ((S1*Z) - (S1*T))/(S1*T)
410 LET E(I) = ABS(E(I))
420 LET A1 = E(I)
421 LET Y2 = T
422 GOSUB 9000
430 IF ABS(A1) < C THEN 450
440 LET A$ = "*****"
450 PRINT X, Y, T, Z, E(I); A$
460 NEXT I
470 PRINT
480 PRINT "                        ABSOLUTE RELATIVE ERROR RANGE"
490 PRINT
500 PRINT
660 FOR I = 1 TO 17
670 IF H > E(I) THEN 700
680 LET H = E(I)
690 GO TO 720
700 IF L <= E(I) THEN 720
710 LET L = E(I)
720 NEXT I
730 PRINT
740 PRINT "          LOWEST ERROR = "; L ; "    HIGHEST ERROR = "; H
750 PRINT
810 PRINT "                                    END TEST"
850 GO TO 9400
9000 IF 1 <= ABS(Y2) THEN 9101
9001 LET S2 = 1
9003 IF ABS(Y2) > 1E-37 THEN 9010
9005 LET P = 1E-37
9007 LET S2 = 1E10
9009 GO TO 9200
9010 LET P = 1
9020 LET P1 = P/10
9030 IF P1 <= ABS(Y2) THEN 9200
9040 LET P = P/10
9050 GO TO 9020
9101 LET S2 = 1
9105 IF ABS(Y2) <= 1E37 THEN 9110
```

```
9106 LET P = 1E38
9107 LET S2 = 1E-10
9109 GO TO 9200
9110 LET P = 10
9120 IF ABS(Y2) < P THEN 9200
9130 LET P = P*10
9140 GO TO 9120
9200 LET Y1 = ABS(Y2)
9210 LET C = (.5E-6)*((P*S2)/(Y1*S2))
9300 RETURN
9400 END
```

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

PROGRAM FILE 106

SECTION 106.0

EXPONENTIATION
X^Y

BEGIN TEST

X	Y	TEST VALUE	SYSTEM VALUE	RELATIVE ERROR
1.00000E+38	.5	1.00000E+19	1.00000E+19	0
1.00000E+38	-1	1.00000E-38	1.00000E-38	0
9.99999E+18	2	9.99998E+37	9.99998E+37	0
100	7	1.00000E+14	1.00000E+14	0
100	-7	1.00000E-14	1.00000E-14	0
10	24	1.00000E+24	1.00000E+24	0
10	-24	1.00000E-24	1.00000E-24	0
2	89	6.18970E+26	6.18970E+26	0
2	66.6	1.11840E+20	1.11840E+20	0
2	16	65536	65536	0
2	-44.3	4.61712E-14	4.61712E-14	0
1.0001	100	1.01005	1.01005	0
1	88	1	1	0
1	-1	1	1	0
.9999	77.777	.992252	.992252	0
1.73715E-11	3	5.24218E-33	5.24218E-33	0
1.00000E-38	0.5	1.00000E-19	1.00000E-19	0

ABSOLUTE RELATIVE ERROR RANGE

```
LOWEST ERROR =  Ø     HIGHEST ERROR =  Ø

END TEST
```

107.0 EXCEPTION TEST FOR TAN FUNCTION AT PI/2

This routine tests for overflow of the tangent function near Pi/2. If overflow occurs, it should be reported, and program terminated (see section 8.5 of BSR X3.60). Pi/2, in the program, is approached on the right and the left by 5 pairs of points, each pair having 6, 7, 8, 14 and 56 digits of precision. With infinite precision, as the point to the left of pi/2 approaches, the TAN function approaches positive infinity. As the points to the right approach pi/2 the TAN function approaches negative infinity. But, given finite precision and roundoff considerations the user should see the evaluation of the tangent function give the same value for all arguments after some point. If there is an overflow then it should be reported and the program suspended. But from the present precision available on machines it is highly unlikely to experience overflow and the program should proceed to termination.

```
**********************
* PROGRAM FILE 107 *
**********************
```

```
10 PRINT "PROGRAM FILE 107"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "  SECTION 107.0: EXCEPTION TEST FOR TAN NEAR PI/2"
60 PRINT
70 PRINT "                                BEGIN TEST"
335 REM 6 DIGIT ARGUMENTS SURROUNDING PI/2
340 LET X1 = TAN(1.57080)
345 LET Y1 = TAN(1.57079)
350 REM 7 DIGIT ARGUMENTS SURROUNDING PI/2
355 LET X2 = TAN(1.570797)
360 LET Y2 = TAN(1.570796)
365 REM 8 DIGIT ARGUMENTS SURROUNDING PI/2
370 LET X3 = TAN(1.5707964)
375 LET Y3 = TAN(1.5707963)
380 REM 14 DIGIT ARGUMENTS SURROUNDING PI/2
385 LET X4 = TAN(1.5707963267949)
390 LET Y4 = TAN(1.5707963267948)
395 REM 56 DIGIT ARGUMENTS SURROUNDING PI/2
400 LET X5 = 1.57079632679489661923132169163975144209858469968756
405 LET Y5 = 1.57079632679489661923132169163975144209858469968755
410 LET X5 = TAN(X5)
415 LET Y5 = TAN(Y5)
420 PRINT "THIS ROUTINE ATTEMPTS TO FORCE AN OVERFLOW FOR THE"
425 PRINT "TAN FUNCTION NEAR PI/2."
430 PRINT "FIVE PAIRS OF CONSTANTS ARE USED AS ARGUMENTS"
435 PRINT "EACH PAIR BRACKETS PI/2 WITHOUT REGARD TO ROUNDING"
```

```
440 PRINT "THE FIVE PAIRS HAVE 6, 7, 8, 14, AND 56 DIGITS RESPECTIVELY"
445 PRINT "ASSUMING INFINITE PRECISION CAPABILITY COLUMN 1 BELOW"
450 PRINT "SHOULD TEND TO + INFINITY AND COLUMN 2 TO - INFINITY"
455 PRINT "FINITE PRECISION ARITHMETIC AND ROUNDOFF WILL GENERATE THE"
456 PRINT "SAME NUMBER IN BOTH COLUMNS AT SOME POINT"
459 PRINT
460 PRINT Y1, X1
465 PRINT Y2, X2
470 PRINT Y3, X3
475 PRINT Y4, X4
480 PRINT Y5, X5
485 PRINT
490 PRINT "IF OVERFLOW OCCURRED ABOVE THEN IT SHOULD HAVE BEEN"
491 PRINT "REPORTED AND PROGRAM TERMINATED. IF NOT THEN IT IS"
495 PRINT "UNLIKELY THAT TAN FUNCTION WILL OVERFLOW FOR ANY"
496 PRINT "CALCULATION."
500 PRINT
510 PRINT "                              END TEST"
515 END
```

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

```
PROGRAM FILE 107


  SECTION 107.0: EXCEPTION TEST FOR TAN NEAR PI/2

                          BEGIN TEST
THIS ROUTINE ATTEMPTS TO FORCE AN OVERFLOW FOR THE
TAN FUNCTION NEAR PI/2.
FIVE PAIRS OF CONSTANTS ARE USED AS ARGUMENTS
EACH PAIR BRACKETS PI/2 WITHOUT REGARD TO ROUNDING
THE FIVE PAIRS HAVE 6, 7, 8, 14, AND 56 DIGITS RESPECTIVELY
ASSUMING INFINITE PRECISION CAPABILITY COLUMN 1 BELOW
SHOULD TEND TO + INFINITY AND COLUMN 2 TO - INFINITY
FINITE PRECISION ARITHMETIC AND ROUNDOFF WILL GENERATE THE
SAME NUMBER IN BOTH COLUMNS AT SOME POINT

 157649        -272120
 3.05163E+6    -1.47320E+6
 2.84819E+7    -1.42409E+7
 2.84819E+7     2.84819E+7
 2.84819E+7     2.84819E+7

IF OVERFLOW OCCURRED ABOVE THEN IT SHOULD HAVE BEEN
REPORTED AND PROGRAM TERMINATED. IF NOT THEN IT IS
```

UNLIKELY THAT TAN FUNCTION WILL OVERFLOW FOR ANY
CALCULATION.

END TEST

108.0 RND FUNCTION WITHOUT RANDOMIZE

The objective of this test is to verify that the implementation-supplied RND function will generate the same sequence of pseudo-random numbers each time a program is run (see section 8 of BSR X3.60). The test program below generates 20 random numbers using the RND function without the RANDOMIZE statement. In order to test that the same sequence of 20 random numbers is regenerated the user should repeat the program several (3 or 4) times. On output, there is a printed message informing the user that 20 random numbers follow the message. After these numbers are printed, another message should inform the user that the same 20 random numbers should occur for each execution of the program. The sample output below only gives the result of two sample executions. The main point of the test is that the sequence of 20 numbers is the same in each execution, and that each number is nonzero, and less than one.

****
* PROGRAM FILE 108 *
****

```
0010 PRINT "PROGRAM FILE 108"
0060 PRINT
0070 PRINT
0080 PRINT
0085 PRINT "SECTION 108.0: RND FUNCTION WITHOUT RANDOMIZE STATEMENT"
0190 PRINT
0200 DIM A(20)
0210 FOR I=1 TO 20
0220 LET A(I)=RND
0230 NEXT I
0240 PRINT "     THE FOLLOWING 20 NUMBERS SHOULD BE THE IMPLEMENTATION-"
0250 PRINT "PREDEFINED SEQUENCE OF PSEUDO-RANDOM NUMBERS OF THIS SYSTEM"
0260 PRINT "AS VALUES FOR ITS RND FUNCTION."
0270 PRINT
0280 FOR I=1 TO 20
0290 PRINT TAB(33);A(I)
0300 NEXT I
0310 PRINT
0320 PRINT "     THE ABOVE SEQUENCE OF PSEUDO-RANDOM NUMBERS SHOULD BE"
0330 PRINT "THE SAME FOR EACH PROGRAM EXECUTION."
0340 END
```

****
* SAMPLE OUTPUT *

****

PROGRAM FILE 108

SECTION 108.0: RND FUNCTION WITHOUT RANDOMIZE STATEMENT

THE FOLLOWING 20 NUMBERS SHOULD BE THE IMPLEMENTATION-PREDEFINED SEQUENCE OF PSEUDO-RANDOM NUMBERS OF THIS SYSTEM AS VALUES FOR ITS RND FUNCTION.

.217873
.696209
.29751
.963794
.463246
.767746
.181667
.159454
6.52568E-2
.495683
.644913
.927201
.67735
.804367
.992458
2.75721E-2
.322263
.731568
.704922
.12663

THE ABOVE SEQUENCE OF PSEUDO-RANDOM NUMBERS SHOULD BE THE SAME FOR EACH PROGRAM EXECUTION.

PROGRAM FILE 108

SECTION 108.0: RND FUNCTION WITHOUT RANDOMIZE STATEMENT

THE FOLLOWING 20 NUMBERS SHOULD BE THE IMPLEMENTATION-PREDEFINED SEQUENCE OF PSEUDO-RANDOM NUMBERS OF THIS SYSTEM AS VALUES FOR ITS RND FUNCTION.

.217873
.696209
.29751

```
.963794
.463246
.767746
.181667
.159454
6.52568E-2
.495683
.644913
.927201
.67735
.804367
.992458
2.75721E-2
.322263
.731568
.704922
.12663
```

THE ABOVE SEQUENCE OF PSEUDO-RANDOM NUMBERS SHOULD BE THE SAME FOR EACH PROGRAM EXECUTION.

109.0 RND FUNCTION WITH RANDOMIZE

The objective of this test is to verify that the RANDOMIZE-statement overrides the implementation-predefined sequence of pseudo-random numbers as values for the RND function (see section 17 of BSR X3.60). This allows different (and unpredictable) sequences each time a given program is executed. In order to test that a different sequence occurs each time the program is executed this program should be run several (3 or 4) times. On output, a message should tell the user that 20 random numbers should follow the message. After the numbers have been generated, another message should indicate that each execution of the program should generate a different set of 20 unpredictable random numbers. Only the results of two executions will be given below.

```
**********************
* PROGRAM FILE 109 *
**********************
```

```
0010 PRINT "PROGRAM FILE 109"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 109.0"
0100 PRINT
0110 PRINT "              (RND FUNCTION WITH THE RANDOMIZE STATEMENT.)"
0150 DIM A(20)
0160 RANDOMIZE
0170 FOR I=1 TO 20
0180 LET A(I)=RND
0190 NEXT I
0200 PRINT "     THE FOLLOWING SEQUENCE OF 20 NUMBERS SHOULD BE AN OVER"
0210 PRINT "RIDE OF THE IMPLEMENTATION-PREDEFINED SEQUENCE OF PSEUDO-"
0220 PRINT "RANDOM NUMBERS OF THIS SYSTEM AS VALUES FOR ITS RND FUNC-"
0230 PRINT "TION."
0240 PRINT
0250 FOR I=1 TO 20
0260 PRINT TAB(33);A(I)
0270 NEXT I
0280 PRINT
0290 PRINT "     THE ABOVE SEQUENCE SHOULD BE DIFFERENT (AND UNPREDICT-"
0300 PRINT "ABLE FOR EACH PROGRAM EXECUTION."
0310 END
```

****

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

PROGRAM FILE 109

                    SECTION 109.0

          (RND FUNCTION WITH THE RANDOMIZE STATEMENT.)
     THE FOLLOWING SEQUENCE OF 20 NUMBERS SHOULD BE AN OVER
RIDE OF THE IMPLEMENTATION-PREDEFINED SEQUENCE OF PSEUDO-
RANDOM NUMBERS OF THIS SYSTEM AS VALUES FOR ITS RND FUNC-
TION.

                         .49927
                         .723266
                         .876503
                         .286454
                         .761588
                         .250402
                         .785724
                         .484855
                         .126906
                         .707897
                         .547312
                         .735257
                         .493621
                         3.21666E-2
                         .862163
                         .12934
                         4.09330E-2
                         .767423
                         .622961
                         7.30995E-2

     THE ABOVE SEQUENCE SHOULD BE DIFFERENT (AND UNPREDICT-
ABLE FOR EACH PROGRAM EXECUTION.


PROGRAM FILE 109

                    SECTION 109.0

          (RND FUNCTION WITH THE RANDOMIZE STATEMENT.)
     THE FOLLOWING SEQUENCE OF 20 NUMBERS SHOULD BE AN OVER
RIDE OF THE IMPLEMENTATION-PREDEFINED SEQUENCE OF PSEUDO-
RANDOM NUMBERS OF THIS SYSTEM AS VALUES FOR ITS RND FUNC-
```

TION.

```
.690698
.228783
.618376
.764632
.790845
3.87759E-2
.455331
1.96282E-2
.657152
.458736
.246176
5.84041E-2
.914067
.265804
.715556
.152839
.160243
.324208
6.69062E-2
.426047
```

THE ABOVE SEQUENCE SHOULD BE DIFFERENT (AND UNPREDICTABLE FOR EACH PROGRAM EXECUTION.

110.0 UNIFORMITY TEST FOR THE RND FUNCTION

The RND function generates the next pseudo-random number in and implementation-supplied sequence of pseudo-random numbers uniformly distributed from 0 to 1 (see section 8 of BSR X3.60). The uniformity of the distribution of the numbers is a statistically measurable quantity. This is the objective of the following test program, in which two statistical "goodness-of-fit" tests are performed. In general goodness of fit tests are used to determine whether an assumed population distribution is consistent with data available. The hypothesis we wish to test is whether the population density is given by the uniform density.

We first test the local behavior of the random number generator by performing 30 statistical experiments. Each experiment consists of sampling 2000 random numbers from the unit interval. Since the RANDOMIZE statement is used each sequence begins with a different seed number for the random nmber generator. We test the uniformity of each sequence by using a chi-square test.

To compute the chi-square value we first divide the unit interval into k subintervals (in our case k=21). If the subintervals are of the same length then the probability of a uniform random number falling into a subinterval is 1/k. We note that there is a rule of thumb for selecting the number of equal subintervals. That is that if n is the sample size (in our case n = 2000) and k is the number of subintervals then we must choose n and k so that n/k >= 5. In our case this is easily satisfied so that our coice of the number of subintervals was satisfactory. The number n/k is the expected frequency of uniform numbers per subinterval. In our test this is approximately 95 per subinterval.

Once the frequency count per subinterval has been computed tne chi-square value for each experiment can computed and this value compared against a table of chi-square values with k-1 degrees of freedom (see, for example Handbook of Mathematical Functions, ed. by M. Abomowitz and I. A. Stegun, U.S. Government Printing Office, 1964, Table 26.8). In our case we have 20 degrees of freedom. For each experiment if the chi-square value is less than the 99-percent entry or greater than the 1-percent entry we reject the experiment as not sufficiently random. If the chi-square value lies between the 99 and 95-percent entries or between the 5 and 1-percent entries the numbers are suspect. Finally, if the chi-square lies between the 95 and 90-percent entries or 10 and 5 percent entries the sequence is considered almost suspect. If the chi-square value is any other number, the hypothesis is accepted that the underlying distribution is uniform locally.

After the 30 chi-square values have been generated these numbers themselves can be tested against the chi-square distribution. This is done by the Kolmogorov-Smirnov test. The output of this test is a set of two numbers. Each must lie within the acceptable range in order for the test to have been successful. The techniques involved are discussed in D. E. Knuth, The Art of Computer Programming, Vol. 2, Addison-Wesley Publishing Company, Reading, Massachusetts (1969). The chi-square cumulative distribution function computed in lines 4900 to 5800 in the program is computed directly from the integral definition using the fact that for an even degree of freedom one can use a table of integrals to determine the iteration formula in line 5680. For example, the user might consult C.R.C Standard Mathematical Tables, Thirteenth Edition, Ed. Robert C. Weast, The

Chemical Rubber Co., Cleveland (1964), p. 317, Formula 354.

```
**********************
* PROGRAM FILE 110 *
**********************
```

```
2 PRINT "PROGRAM FILE 110"
5 PRINT
6 PRINT
7 PRINT
10 PRINT "   SECTION 110.0: UNIFORMITY TEST FO THE RND FUNCTION"
15 PRINT
20 PRINT "                                BEGIN TEST"
30 PRINT
50 DIM Y(21),V(30),F(30)
100 PRINT "EXPERIMENT", "CHI-SQUARE", "EVALUATION"
110 PRINT
150 REM PERFORM 30 EXPERIMENTS OF 2000 SAMPLES EACH
200 REM OF PSEUDO-RANDOM NUMBERS THEN COMPUTE
250 REM AND TEST CHI-SQ STATISTIC FOR EACH EXPERIMENT
350 LET R1 = 0
400 LET S1 = 0
450 LET S2 = 0
500 LET A  = 0
550 FOR E = 1 TO 30
600 RANDOMIZE
650 REM INITIALIZE OBSERVED COUNTS AT 0 FOR A PARTITION OF THE
700 REM UNIT INTERVAL INTO 21 CELLS
800 FOR I = 1 TO 21
850 LET Y(I) = 0
900 NEXT I
950 REM OBTAIN FREQUENCY COUNTS FOR 2000 RANDOM NUMBERS
1050 FOR I = 1 TO 2000
1100 LET X = RND
1150 LET R = INT(21*X) + 1
1200 LET Y(R) = Y(R) + 1
1250 NEXT I
1300 REM COMPUTE CHI-SQ STATISTIC FOR EACH EXPERIMENT
1400 LET V(E) = 0
1450 LET S = 0
1500 FOR I = 1 TO 21
1550 LET S = S + (21*Y(I)*Y(I))
1600 NEXT I
1650 LET V(E) = (1/2000)*S - 2000
1700 REM TEST CHI-SQ STATISTIC FOR EACH EXPERIMENT
1800 IF V(E) >= 8.260 THEN 2000
1850 LET R1 = R1 + 1
1900 LET F$ = "REJECT"
1950 GO TO 3150
```

```
2000 IF V(E) >= 10.85 THEN 2200
2050 LET S1 = S1 + 1
2100 LET F$ = "SUSPECT"
2150 GO TO 3150
2200 IF V(E) >= 12.44 THEN 2400
2250 LET S2 = S2 + 1
2300 LET F$ = "ALMOST SUSPECT"
2350 GO TO 3150
2400 IF V(E) > 28.41 THEN 2600
2450 LET A = A + 1
2500 LET F$ = "ACCEPT"
2550 GO TO 3150
2600 IF V(E) >31.41 THEN 2800
2650 LET S2 = S2 + 1
2700 LET F$ = "ALMOST SUSPECT"
2750 GO TO 3150
2800 IF V(E) > 37.57 THEN 3000
2850 LET S1 = S1 + 1
2900 LET F$ = "SUSPECT"
2950 GO TO 3150
3000 LET F$ = "REJECT"
3050 LET R1 = R1 + 1
3100 REM PRINT EVALUATION
3150 PRINT E, V(E), F$
3200 NEXT E
3250 PRINT
3300 PRINT "SUMMARY EVALUATION STATISTICS"
3350 PRINT
3400 PRINT "NUMBER OF REJECTS = ";R1
3450 PRINT "NUMBER OF SUSPECTS = ";S1
3500 PRINT "NUMBER OF ALMOST SUSPECTS = ";S2
3550 PRINT "NUMBER OF ACCEPTS = ";A
3600 PRINT
3650 PRINT "THE GENERATOR SHOWS POOR LOCAL RANDOMNESS IF THERE"
3660 PRINT "ARE ANY REJECTS."
3700 PRINT "IF THE NUMBER OF ACCEPTS DOES NOT OUTWEIGH THE OTHER"
3750 PRINT "CATEGORIES THEN THIS SYSTEM IS USING A POOR RND FUNCTION"
3760 PRINT "BUT FINAL REJECTION DEPENDS ON THE RESULTS OF THE"
3770 PRINT "FINAL STATISTICS FOR UNIFORMITY."
3900 REM COMPUTING KOLMOGOROV-SMIRNOV STATISTIC TO TEST GLOBAL
3950 REM UNIFORMITY OF THE PSEUDO-RANDOM NUMBER GENERATOR
4000 REM SORT CHI-SQ VALUES V(E)
4050 LET Z = 0
4100 FOR I = 1 TO 29
4150 IF V(I) <= V(I+1) THEN 4400
4200 LET Q = V(I)
4250 LET V(I) = V(I+1)
4300 LET V(I+1) = Q
4350 LET Z = 1
4400 NEXT I
4450 IF Z > 0.5 THEN 4050
4500 REM COMPUTE TWO KOLMOGOROV- SMIRNOV STATISTICS K+ AND K-.
4600 REM CALL K+ BY K1 AND K- BY K2. BEGIN BY COMPUTING THE
4700 REM PROBABILITIES FOR V(1),...,V(30) FROM THE CUMULATIVE
4750 REM OF THE CHI-SQ DISTRIBUTION. THIS IS DONE BELOW FOR THE
4800 REM CASE OF AN EVEN DEGREE OF FREEDOM N.
4900 LET N = 20
```

```
4950 LET M = N/2
5000 LET K = M - 1
5010 REM COMPUTING GAMMA FUNCTION OF M OR (M-1)-FACTORIAL.
5050 LET F1 = 1
5100 IF K = 0 THEN 5300
5150 FOR I = 1 TO K
5200 LET F1 = I * F1
5250 NEXT I
5260 REM NORMALIZING COEFFICIENT
5300 LET C = 1/((2M)*F1)
5350 REM COMPUTING INTEGRAL ITERATIVELY
5400 FOR I = 1 TO 30
5450 LET X2 = V(I)
5500 LET I1 = 2 * (1 - EXP(-0.5*X2))
5550 IF K = 0 THEN 5750
5600 FOR J = 1 TO K
5650 LET I1 = 2*J*I1 - 2*(X2J)*EXP(-0.5*X2)
5700 NEXT J
5750 LET F(I) = C * I1
5800 NEXT I
5900 REM LET M1 = MAX OF J/30 - F(J) FOR J = 1 TO 30 AND M2 =
5950 REM MAX OF F(J) - J/30 FOR J = 1 TO 30. COMPUTE K+ = K1 =
6050 REM SQR(30)*M1 AND K- = K2 = SQR(30)*M2
6150 LET M1 = (1/30) - F(1)
6200 LET M2 = F(1)
6250 FOR J = 2 TO 30
6300 IF M1 >= (J/30) - F(J) THEN 6400
6350 LET M1 = (J/30) - F(J)
6400 IF M2 >= F(J) - ((J-1)/30) THEN 6500
6450 LET M2 = F(J) - ((J-1)/30)
6500 NEXT J
6550 LET K1 = SQR(30) * M1
6600 LET K2 = SQR(30) * M2
6650 REM TEST KOLMOGOROV-SMIRNOV STATISTIC
6750 PRINT
6800 PRINT "KOLMOGOROV-SMIRNOV STATISTIC TEST FOR UNIFORMITY"
6850 PRINT
6860 LET Q$ = " "
6900 IF K1 >= 0.04354 THEN 7050
6950 LET Q$ = "REJECT"
7000 GO TO 7550
7050 IF K1 >= 0.1351 THEN 7200
7100 LET Q$ = "SUSPECT"
7150 GO TO 7550
7200 IF K1 > 1.916 THEN 7350
7250 LET Q$ = "ACCEPT"
7300 GO TO 7550
7350 IF K1 > 1.4801 THEN 7500
7400 LET Q$ = "SUSPECT"
7450 GO TO 7550
7500 LET Q$ = "REJECT"
7550 PRINT "K+ = "; K1, Q$
7560 LET Q$ = " "
7600 IF K2 >= 0.04354 THEN 7750
7650 LET Q$ = "REJECT"
7700 GO TO 8250
7750 IF K2 >= 0.1351 THEN 7900
```

```
7800 LET Q$ = "SUSPECT"
7850 GO TO 8250
7900 IF K2 >1.916 THEN 8050
7950 LET Q$ = "ACCEPT"
8000 GO TO 8250
8050 IF K2 > 1.4801 THEN 8200
8100 LET Q$ = "SUSPECT"
8150 GO TO 8250
8200 LET Q$ = "REJECT"
8250 PRINT "K- = "; K2, Q$
8300 PRINT
8350 PRINT "IF EITHER STATISTIC IS LESS THAN ACCEPTABLE,"
8400 PRINT "THIS INDICATES GLOBAL NONRANDOMNESS."
8500 PRINT
8600 PRINT "                                END TEST"
9000 END
```

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

```
PROGRAM FILE 110


   SECTION 110.0: UNIFORMITY TEST FOR THE RND FUNCTION

                          BEGIN TEST

EXPERIMENT    CHI-SQUARE    EVALUATION

 1              19.801       ACCEPT
 2              10.372       SUSPECT
 3              12.157       ALMOST SUSPECT
 4              22.657       ACCEPT
 5              10.624       SUSPECT
 6              10.246       SUSPECT
 7              16.504       ACCEPT
 8              22.72        ACCEPT
 9              14.656       ACCEPT
 10             14.593       ACCEPT
 11             18.919       ACCEPT
 12             22.678       ACCEPT
 13             10.939       ALMOST SUSPECT
 14             30.259       ALMOST SUSPECT
 15             30.259       ALMOST SUSPECT
 16             15.139       ACCEPT
 17             17.806       ACCEPT
 18             16.315       ACCEPT
```

19	7.76801	REJECT
20	17.617	ACCEPT
21	24.211	ACCEPT
22	20.053	ACCEPT
23	18.94	ACCEPT
24	32.653	SUSPECT
25	49.894	REJECT
26	19.885	ACCEPT
27	19.717	ACCEPT
28	9.658	SUSPECT
29	25.765	ACCEPT
30	8.79701	SUSPECT

SUMMARY EVALUATION STATISTICS

NUMBER OF REJECTS = 2
NUMBER OF SUSPECTS = 6
NUMBER OF ALMOST SUSPECTS = 4
NUMBER OF ACCEPTS = 18

THE GENERATOR SHOWS POOR LOCAL RANDOMNESS IF THERE
ARE ANY REJECTS.
IF THE NUMBER OF ACCEPTS DOES NOT OUTWEIGH THE OTHER
CATEGORIES THEN THIS SYSTEM IS USING A POOR RND FUNCTION
BUT FINAL REJECTION DEPENDS ON THE RESULTS OF THE
FINAL STATISTICS FOR UNIFORMITY.

KOLMOGOROV-SMIRNOV STATISTIC TEST FOR UNIFORMITY

K+ =	.992047	ACCEPT
K- =	.370091	ACCEPT

IF EITHER STATISTIC IS LESS THAN ACCEPTABLE,
THIS INDICATES GLOBAL NONRANDOMNESS.

END TEST

111.0 USER DEFINED FUNCTIONS WITH A PARAMETER LIST

The next two sections test whether the implementation allows user-defined functions. They furthermore verify the forms that such functions can take. The reader is referred to section 16 of BSR X3.60. The objective of this first section is to test the basic forms a definition can take when the defining function uses a parameter-list as follows:

```
DEF FNx(parameter)=expression
```

where x is a single letter and a parameter is a simple-numeric- variable.

111.1 Defining Expression Uses Only the Parameter

The primary objective of this subtest is to verify that the parameter in the parameter-list can be used by itself to form the expression needed in defining the function. That is, we test that the implementation allows the following form of a statement for defining a function:

```
DEF FNA(X)=X*X
```

where X is a the parameter in the parameter-list. A secondary objective is to test some of the forms an expression can have when introduced into the parameter list at the time of a function reference. In particular the expression in the argument list for the function references that are being tested are: (1) a constant, (2) a simple numeric variable, (3) a simple expression of two or more terms, and (4) an expression consisting of another function reference. On output, the test informs the user of any evaluation failure by the implementation. For each failure a message is supplied indicating the nature of the failure.

111.2 The Defining Expression Uses Other Parameters

The objective of this subtest is to verify that both the parameter in the parameter-list and other variable(s) can be used to form the expression needed in defining the function. That is, we test whether the implementation allows the following form of a statement in defining a function:

```
DEF FNA(X)=A*X+B
```

where X is the parameter in the parameter-list. On output, there should be a message indicating whether the implementation passed or failed the test.

111.3 The Defining Expression Uses Constants

The objective of this subtest is to verify that both the parameter in the parameter-list and constants can be used to form the expression used to define the function. In particular, we test whether the implementation allows the following form of a statement for defining a function:

```
DEF FNA(X)=X^4-1
```

where X is the parameter in the parameter-list. On output, there should be a

message indicating whether the implementation passed or failed the test.

111.4 The Defining Expression References Another Defined Function

The objective of this subtest is to verify that a preceding definition can be used in a succeeding definition (i.e., a definition in a lower line number should be allowed in another definition if the latter is in a higher line number). This test also verifies that the parameter appearing in the parameter-list of a function definition is local to that definition (i.e., it is distinct from any variable with the same name outside of the function definition. However the variables in the defining expression that do not appear in the parameter-list are the variables of the same name outside the function definition). The program below verifies that the implementation will properly evaluate several definitions in the same program as follows:

```
DEF FNA(Z)=Z^2-4
DEF FNB(Z)=Z*X+FNA(A^2+B)
```

where Z is the parameter for both parameter-lists. On output, there should be a message indicating whether the implementation passed or failed the test.

```
**********************
* PROGRAM FILE 111 *
**********************
```

```
0010 PRINT "PROGRAM FILE 111"
0020 PRINT
0030 PRINT
0040 PRINT
0050 PRINT "                                SECTION 111.0."
0090 PRINT "      TESTING  USER-DEFINED FUNCTIONS WITH PARAMETER LISTS"
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "*****NOTE:  THIS UNIT IS ALSO A PRELIMINARY INTRODUCTION TO"
0140 PRINT "FUNCTIONS OF EXPRESSIONS.  MORE EXTENSIVE CASES WILL OCCUR"
0150 PRINT "IN A LATER UNIT.*****"
0200 PRINT
0210 PRINT
0220 PRINT
0230 PRINT "                          SECTION 111.1"
0240 PRINT
0250 PRINT "(THE EXPRESSION OF THE DEFINITION USING ONLY THE PARAMENTER"
0260 PRINT " OF THE PARAMENTER-LIST.)"
0270 PRINT
0280 PRINT
0290 PRINT "                                BEGIN TEST."
0300 PRINT
```

```
0310 DEF FNA(X)=X*X
0320 DEF FNB(Y)=Y*Y*Y
0330 LET A=5
0340 LET B=4
0350 LET C=3
0360 LET E=0
0370 LET X=FNA(12)
0380 LET Y=FNA(A)
0390 LET Z=FNA(2*A^3+4*B+C)
0400 LET W=FNA(FNB(2)+B^3)
0410 LET F=1
0420 IF X=144 THEN 440
0430 GOSUB 540
0440 LET F=2
0450 IF Y=25 THEN 470
0460 GOSUB 540
0470 LET F=3
0480 IF Z=72361 THEN 500
0490 GOSUB 540
0500 LET F=4
0510 IF W=5184 THEN 680
0520 GOSUB 540
0530 GOTO 680
0540 LET E=1
0550 ON F GOTO 560,580,600,620
0560 LET A$="A CONSTANT"
0570 GOTO 630
0580 LET A$="A VARIABLE"
0590 GOTO 630
0600 LET A$="CONS. AND VARS."
0610 GOTO 630
0620 LET A$="OTHER DEF FNX USED"
0630 PRINT "SYSTEM FAILED TO EXECUTE PROPERLY WHEN THE EXPRESSION"
0640 PRINT "IN THE ARGUMENT-LIST FOR THE FUNCTION REFERENCE WAS"
0650 PRINT A$;"."
0660 PRINT
0670 RETURN
0680 IF E=0 THEN 710
0690 PRINT "SYSTEM FAILED TEST, NOTE THE ABOVE REASONS."
0700 GOTO 720
0710 PRINT "SYSTEM PASSED TEST."
0720 PRINT
0730 PRINT "                                    END TEST."
0740 PRINT
1060 PRINT
1070 PRINT
1080 PRINT
1090 PRINT "                          SECTION 111.2"
1100 PRINT
1110 PRINT " (THE EXPRESSION OF THE DEFINITION USING THE PARAMETER OF"
1120 PRINT "  THE PARAMENTER-LIST AND OTHER VARIABLES.)"
1130 PRINT
1140 PRINT
1150 PRINT "                                    BEGIN TEST."
1160 PRINT
1170 DEF FNC(A)=A*A-B+C
1180 LET B=-15
```

```
1190 LET C=-35
1200 LET X=FNC(13)
1210 IF X=149 THEN 1240
1220 PRINT "SYSTEM FAILED TEST."
1230 GOTO 1250
1240 PRINT "SYSTEM PASSED TEST."
1250 PRINT
1260 PRINT "                                    END TEST."
1270 PRINT
2060 PRINT
2070 PRINT
2080 PRINT
2090 PRINT "                          SECTION 111.3"
2100 PRINT
2110 PRINT " (THE EXPRESSION OF THE DEFINITION USING THE PARAMENTER OF"
2120 PRINT "  THE PARAMETER-LIST AND CONSTANTS.)"
2130 PRINT
2140 PRINT
2150 PRINT "                                  BEGIN TEST."
2160 PRINT
2170 DEF FND(Y)=Y^4-16
2180 LET A=4
2190 LET B=2
2200 LET Y=FND(A^3-3*A^2*B+3*A*B^2-B^3)
2210 IF Y=4080 THEN 2240
2220 PRINT "SYSTEM FAILED TEST."
2230 GOTO 2250
2240 PRINT "SYSTEM PASSED TEST."
2250 PRINT
2260 PRINT "                                    END TEST."
2270 PRINT
3060 PRINT
3070 PRINT
3080 PRINT
3090 PRINT "                          SECTION 111.4"
3100 PRINT
3110 PRINT " (THE EXPRESSION OF THE DEFINITION USING THE PARAMENTER OF"
3120 PRINT "  THE PARAMENTER-LIST AND THE REFERENCE OF ANOTHER DEFINI-"
3130 PRINT "  TION.)"
3140 PRINT
3150 PRINT
3160 PRINT "                                  BEGIN TEST."
3170 PRINT
3180 DEF FNE(Z)=Z^2-4
3190 DEF FNF(Z)=Z*X+FNE(A^2+B)
3200 LET A=9
3210 LET B=19
3220 LET X=3
3230 LET Y=2
3240 LET Z=FNF(X^3-3*X^2*Y+3*X*Y^2-Y^3)
3250 IF Z=9999 THEN 3280
3260 PRINT "SYSTEM FAILED TEST."
3270 GOTO 3290
3280 PRINT "SYSTEM PASSED TEST."
3290 PRINT
3300 PRINT "                                    END TEST."
3310 PRINT
```

```
3320 END
```

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

```
PROGRAM FILE 111

                    SECTION 111.0.
      TESTING  USER-DEFINED FUNCTIONS WITH PARAMETER LISTS

*****NOTE:  THIS UNIT IS ALSO A PRELIMINARY INTRODUCTION TO
FUNCTIONS OF EXPRESSIONS.  MORE EXTENSIVE CASES WILL OCCUR
IN A LATER UNIT.*****

                    SECTION 111.1

(THE EXPRESSION OF THE DEFINITION USING ONLY THE PARAMENTER
 OF THE PARAMENTER-LIST.)

                    BEGIN TEST.

SYSTEM PASSED TEST.

                    END TEST.

                    SECTION 111.2

 (THE EXPRESSION OF THE DEFINITION USING THE PARAMETER OF
  THE PARAMENTER-LIST AND OTHER VARIABLES.)

                    BEGIN TEST.

SYSTEM PASSED TEST.

                    END TEST.
```

SECTION 111.3

(THE EXPRESSION OF THE DEFINITION USING THE PARAMENTER OF THE PARAMETER-LIST AND CONSTANTS.)

BEGIN TEST.

SYSTEM PASSED TEST.

END TEST.

SECTION 111.4

(THE EXPRESSION OF THE DEFINITION USING THE PARAMENTER OF THE PARAMENTER-LIST AND THE REFERENCE OF ANOTHER DEFINITION.)

BEGIN TEST.

SYSTEM PASSED TEST.

END TEST.

112.0 TESTING ALL POSSIBLE PARAMETRIZED USER DEFINED FUNCTION SPECIFICATIONS

The objective of this test is to verify that there are a total of 26 possible user defined function definitions permitted in a single program (see section 16.2 of BSR X3.60). This code tests whether the implementation allows the following general syntax form for a user defined function statement:

DEF FNx(parameter)=expression

where x is a single letter (that is, one of the 26 alphabetic letters) and a parameter is a simple-numeric-variable. The test verifies this by using 26 def-statements defined in order from A through Z in lines 150 to 400 of the source code. When called in the program, the values of the 26 defined functions should decrease in value in the following order 4, 3, 2, 1, ..., -21 and their total sum should be -221. On output, there should be a message informing the user whether the implementation passed or failed the test.

****
* PROGRAM FILE 112 *
****

```
0010 PRINT "PROGRAM FILE 112"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "        SECTION 112: TESTING ALL 26 POSSIBLE DEFINITIONS"
0095 PRINT "                     FOR DEFINED FUNCTIONS"
0100 PRINT
0110 PRINT
0120 PRINT
0130 PRINT "                                  BEGIN TEST."
0140 PRINT
0150 DEF FNA(X)=X/7-7
0160 DEF FNB(X)=X/8-8
0170 DEF FNC(X)=X/9-9
0180 DEF FND(X)=X/10-10
0190 DEF FNE(X)=X/11-11
0200 DEF FNF(X)=X/12-12
0210 DEF FNG(X)=X/13-13
0220 DEF FNH(X)=X/14-14
0230 DEF FNI(X)=X/15-15
0240 DEF FNJ(X)=X/16-16
0250 DEF FNK(X)=X/17-17
0260 DEF FNL(X)=X/18-18
0270 DEF FNM(X)=X/19-19
0280 DEF FNN(X)=X/20-20
```

```
0290 DEF FNO(X)=X/21-21
0300 DEF FNP(X)=X/22-22
0310 DEF FNQ(X)=X/23-23
0320 DEF FNR(X)=X/24-24
0330 DEF FNS(X)=X/25-25
0340 DEF FNT(X)=X/26-26
0350 DEF FNU(X)=X/27-27
0360 DEF FNV(X)=X/28-28
0370 DEF FNW(X)=X/29-29
0380 DEF FNX(X)=X/30-30
0390 DEF FNY(X)=X/31-31
0400 DEF FNZ(X)=X/32-32
0410 DIM A(26)
0420 LET A(1)=FNA(77)
0430 LET A(2)=FNB(88)
0440 LET A(3)=FNC(99)
0450 LET A(4)=FND(110)
0460 LET A(5)=FNE(121)
0470 LET A(6)=FNF(132)
0480 LET A(7)=FNG(143)
0490 LET A(8)=FNH(154)
0500 LET A(9)=FNI(165)
0510 LET A(10)=FNJ(176)
0520 LET A(11)=FNK(187)
0530 LET A(12)=FNL(198)
0540 LET A(13)=FNM(209)
0550 LET A(14)=FNN(220)
0560 LET A(15)=FNO(231)
0570 LET A(16)=FNP(242)
0580 LET A(17)=FNQ(253)
0590 LET A(18)=FNR(264)
0600 LET A(19)=FNS(275)
0610 LET A(20)=FNT(286)
0620 LET A(21)=FNU(297)
0630 LET A(22)=FNV(308)
0640 LET A(23)=FNW(319)
0650 LET A(24)=FNX(330)
0660 LET A(25)=FNY(341)
0670 LET A(26)=FNZ(352)
0680 LET S=0
0690 FOR I=1 TO 26
0700 LET S=S+A(I)
0710 NEXT I
0720 IF S=-221 THEN 750
0730 PRINT "SYSTEM FAILED TEST."
0740 GOTO 760
0750 PRINT "SYSTEM PASSED TEST."
0760 PRINT
0770 PRINT "                                   END TEST."
0780 PRINT
0790 END
```

****

* SAMPLE OUTPUT *
****

```
PROGRAM FILE 112

        SECTION 112: TESTING ALL 26 POSSIBLE DEFINITIONS
                     FOR DEFINED FUNCTIONS

                                BEGIN TEST.

SYSTEM PASSED TEST.

                                END TEST.
```

113.0 USER DEFINED FUNCTIONS WITHOUT A PARAMETER LIST

This section tests some of the forms that expressions can have in the definition of a user defined function without a parameter list (see section 16 of BSR X3.60). We will execute expressions of the general form:

```
DEF FNx=expression
```

where x is a single letter.

113.1 The Definition Uses Only a Constant

This subtest verifies that the expression of a nonparametrized user defined function can be a constant. In particular we define:

```
DEF FNx=3.14159
```

where x is a single letter. On output, there should be a message to the user indicating whether the implementation passed or failed the test.

113.2 The Definition Uses a Variable

This subtest verifies that the expression of a nonparametrized user defined function can be a variable. In fact the test determines whether an implementation allows a DEF statement of the following form:

```
DEF FNx=Z*Z
```

where x is a single letter. On output, there should be a message indicating whether the implementation passed of failed the test.

113.3 The Definition Uses Both Constants and Variables

This subtest verifies that the expression in the definition can use both constants and variables together in an expression of the form:

```
DEF FNx=A^2+8*A*B+16
```

where x is a single letter. On output, there should be a message indicating whether the implementation passed or failed the test.

113.4 The Definition References Another Defined Function

This subtest verifies that the expression of a nonparametrized definition can contain a reference to another defined function. The following expressions are used in this test:

```
DEF FND=X^4-1
DEF FNE=FND+45
```

where D and E are the single letters which make the distinguishing factor between the two definitions. On output, there should be a message indicating whether the implementation passed or failed the test.

```
**********************
* PROGRAM FILE 113 *
**********************
```

```
1050 PRINT "PROGRAM FILE 113"
1060 PRINT
1070 PRINT
1080 PRINT
1085 PRINT " SECTION 113.0: DEFINED FUNCTIONS WITHOUT A PARAMETER LIST"
1100 PRINT
1110 PRINT
1120 PRINT
1130 PRINT "                          SECTION 113.1"
1140 PRINT
1150 PRINT "      (THE EXPRESSION OF THE DEFINITION USING A CONSTANT.)"
1160 PRINT
1170 PRINT
1180 PRINT "                                BEGIN TEST."
1190 PRINT
1200 DEF FNA=3.14159
1210 LET A=10
1220 LET Y=FNA*A^2
1230 LET X=(Y-314.159)/314.159
1240 IF X < 3.1831E-6 THEN 1270
1250 PRINT "SYSTEM FAILED TEST."
1260 GOTO 1280
1270 PRINT "SYSTEM PASSED TEST."
1280 PRINT
1290 PRINT "                                END TEST."
1300 PRINT
2070 PRINT
2080 PRINT
2090 PRINT "                          SECTION 113.2"
2100 PRINT
2110 PRINT "      (THE EXPRESSION OF THE DEFINITION USING A VARIABLE.)"
2120 PRINT
2130 PRINT
2140 PRINT "                                BEGIN TEST."
2150 PRINT
2160 DEF FNB=X*X
2170 LET X=31
2180 LET Y=FNB+38
2190 IF Y=999 THEN 2220
2200 PRINT "SYSTEM FAILED TEST."
2210 GOTO 2230
2220 PRINT "SYSTEM PASSED TEST."
2230 PRINT
2240 PRINT "                                END TEST."
2250 PRINT
3070 PRINT
3080 PRINT
3090 PRINT "                          SECTION 113.3"
3100 PRINT
```

```
3110 PRINT "(THE EXPRESSION OF THE DEFINITION USING BOTH CONSTANTS AND"
3120 PRINT " VARIABLES TOGETHER.)"
3130 PRINT
3140 PRINT
3150 PRINT "                                   BEGIN TEST."
3160 PRINT
3170 DEF FNC=A^2+8*A*B+16
3180 LET A=5
3190 LET B=4
3200 LET C=FNC^2-1
3210 IF C=40400 THEN 3240
3220 PRINT "SYSTEM FAILED TEST."
3230 GOTO 3250
3240 PRINT "SYSTEM PASSED TEST."
3250 PRINT
3260 PRINT "                                    END TEST."
3270 PRINT
4070 PRINT
4080 PRINT
4090 PRINT "                          SECTION 113.4"
4100 PRINT
4110 PRINT "(THE EXPRESSION OF THE DEFINITION USING THE REFERENCE OF"
4120 PRINT " ANOTHER DEFINITION FOR ONE OF ITS TERMS.)"
4130 PRINT
4140 PRINT
4150 PRINT "                                   BEGIN TEST."
4160 PRINT
4170 DEF FND=X^4-1
4180 DEF FNE=FND+45
4190 LET X=2
4200 LET Y=FNE+602
4210 IF Y=3660 THEN 4240
4220 PRINT "SYSTEM FAILED TEST."
4230 GOTO 4250
4240 PRINT "SYSTEM PASSED TEST."
4250 PRINT
4260 PRINT "                                    END TEST."
4270 PRINT
4280 END
```

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

```
PROGRAM FILE 113


 SECTION 113.0: DEFINED FUNCTIONS WITHOUT A PARAMETER LIST
```

SECTION 113.1

(THE EXPRESSION OF THE DEFINITION USING A CONSTANT.)

BEGIN TEST.

SYSTEM PASSED TEST.

END TEST.

SECTION 113.2

(THE EXPRESSION OF THE DEFINITION USING A VARIABLE.)

BEGIN TEST.

SYSTEM PASSED TEST.

END TEST.

SECTION 113.3

(THE EXPRESSION OF THE DEFINITION USING BOTH CONSTANTS AND VARIABLES TOGETHER.)

BEGIN TEST.

SYSTEM PASSED TEST.

END TEST.

SECTION 113.4

(THE EXPRESSION OF THE DEFINITION USING THE REFERENCE OF ANOTHER DEFINITION FOR ONE OF ITS TERMS.)

BEGIN TEST.

SYSTEM PASSED TEST.

END TEST.

114.0 USER DEFINED FUNCTION DIAGNOSTICS - THE SAME FUNCTION IS DEFINED MORE THAN ONCE

The objective of this test is to show that the implementation will recognize that defining the same function more than once in the same program is semantically meaningless (see section 16.4 of BSR X3.60). The processor should provide a diagnostic, although this error is not considered an exception. In order to test this, the same function is defined both in line 340 andd 350 of the program below. The user should look for a diagnostic noting the double definition. However, if one of the function definitions is ignored by the implementation, then the test program will produce an error message indicating test failure.

```
**********************
* PROGRAM FILE 114 *
**********************
```

```
0010 PRINT "PROGRAM FILE 114"
0240 PRINT
0250 PRINT
0260 PRINT
0270 PRINT "                          SECTION 114.0"
0280 PRINT
0290 PRINT "             (THE SAME FUNCTION IS DEFINED MORE THAN ONCE.)"
0300 PRINT
0310 PRINT
0320 PRINT "                                   BEGIN TEST."
0330 PRINT
0340 DEF FNA(X)=X^4-1
0350 DEF FNA(X)=A*X+B
0360 LET A=5
0370 LET B=6
0380 LET Y=FNA(2)
0390 PRINT "     BY THE APPEARANCE OF THIS STATEMENT, IT IS VERIFIED"
0400 PRINT "THAT THE SYSTEM HAS FAILED THE TEST."
0410 PRINT
0420 PRINT Y
0430 PRINT
0440 PRINT "THE ABOVE VALUE SHOULD NOT HAVE BEEN EVALUATED."
0450 PRINT "                                    END TEST."
0460 PRINT
0470 END
```

```
******************
```

* SAMPLE OUTPUT *
****

```
?FUNCTION DEFINED TWICE IN LINE 350
```

115.0 A USER DEFINED FUNCTION IS REFERENCED INSIDE ITS OWN DEFINITION

The objective of this test is to verify that the implementation will recognize the referencing of a function within its own definition as semantically nonmeaningful and provide a diagnostic (see section 16.4 of BSR X3.60). On output there should be some form of implementation defined diagnostic. However, if the system ignores the illegal definition then a message within the program indicates test failure.

****
* PROGRAM FILE 115 *
****

```
0010 PRINT "PROGRAM FILE 115"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 115.0"
0100 PRINT
0110 PRINT "        (A FUNCTION IS REFERENCED INSIDE ITS DEFINITION.)"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 DEF FNB(X)=X*X+FNB(2)
0170 LET Y=FNB(2)
0180 PRINT "     BY THE APPEARANCE OF THIS STATEMENT, IT IS VERIFIED"
0190 PRINT "THAT THE SYSTEM HAS FAILED THE TEST."
0200 PRINT
0210 PRINT Y
0220 PRINT
0230 PRINT "THE ABOVE VALUE SHOULD NOT HAVE BEEN EVALUATED."
0240 PRINT
0250 PRINT "                                   END TEST."
0260 PRINT
0270 END
```

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

PROGRAM FILE 115

SECTION 115.0

(A FUNCTION IS REFERENCED INSIDE ITS DEFINITION.)

BEGIN TEST.

?FUNCTION CALLS ITSELF IN LINE 170

116.0 SYNTAX DIAGNOSTIC - ARGUMENT LIST USED INCORRECTLY WITH A DEF FUNCTION

This test verifies that the implementation diagnoses an incorrectly used argument list in a "defined" function call statement (see section 16.2 of BSR X3.60). This is a test of syntax specification for user-defined functions. The only output should be a message pinpointing an improper argument list in line 170.

****
* PROGRAM FILE 116 *
****

```
0010 PRINT "PROGRAM FILE 116"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 116.0"
0100 PRINT
0110 PRINT "   ARGUMENT LIST USED INCORRECTLY WITH A DEF FUNCTION"
0120 PRINT
0130 PRINT
0140 PRINT "                                BEGIN TEST."
0150 PRINT
0160 DEF FNA=3.14159
0170 LET Y=FNA(3.14159)
0180 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0190 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0200 PRINT Y
0210 PRINT
0220 PRINT "                                 END TEST."
0230 PRINT
0240 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 170
```

117.0 SYNTAX DIAGNOSTIC - NO ARGUMENT LIST FOR THE ABS FUNCTION

This test determines whether not using an argument with the ABS function causes a syntax diagnostic (see sections 7 and 8 of BSR X3.60). The ABS function appears in an expression on line 210 without its argument. On output, there should be some form of implementation defined diagnostic indicating the error. However if the illegal statement is ignored the test is structured to generate an error message to the user.

****
* PROGRAM FILE 117 *
****

```
0010 PRINT "PROGRAM FILE 117"
0020 PRINT
0030 PRINT
0040 PRINT
0140 PRINT "                          SECTION 117.0"
0150 PRINT
0160 PRINT "          INCORRECT ARGUMENT LIST WITH THE ABS FUNCTION"
0170 PRINT
0180 PRINT
0190 PRINT "                                  BEGIN TEST."
0200 PRINT
0210 LET Y=ABS
0220 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0230 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0240 PRINT Y
0250 PRINT
0260 PRINT "                                   END TEST."
0270 PRINT
0280 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 210
```

118.0 SYNTAX DIAGNOSTIC - NO ARGUMENT LIST WITH ATN

This test determines whether the processor will provide a syntax diagnostic when ATN has an incorrect number of arguments on line 160 (see section 7 and 8 of BSR X3.60). This particular reference to ATN contains no arguments. If the system does not provide a diagnostic and ignores the line then the output will provide a message to the user.

****
* PROGRAM FILE 118 *
****

```
0010 PRINT "PROGRAM FILE 118"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                         SECTION 118.0"
0100 PRINT
0110 PRINT "            NO ARGUMENT LIST WITH THE ATN FUNCTION"
0120 PRINT
0130 PRINT
0140 PRINT "                                 BEGIN TEST."
0150 PRINT
0160 LET Y=ATN
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                   END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160
```

119.0 SYNTAX DIAGNOSTIC - NO ARGUMENT LIST WITH COS

This section tests whether the processor will provide a syntax diagnostic when COS has no argumment in line 160 (see sections 7 and 8 of BSR X3.60). If the statement is ignored then the program will provide a message to the user.

****
* PROGRAM FILE 119 *
****

```
0010 PRINT "PROGRAM FILE 119"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 119.0"
0100 PRINT
0110 PRINT "          NO ARGUMENT LIST WITH COS FUNCTION"
0120 PRINT
0130 PRINT
0140 PRINT "                                BEGIN TEST."
0150 PRINT
0160 LET Y=COS
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                 END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160
```

120.0 SYNTAX DIAGNOSTIC - NO ARGUMENT LIST WITH EXP

This section tests whether the processor will provide a syntax diagnostic when EXP has no arguments in line 160 (see sections 7 and 8 of BSR X3.60). If the statement is ignored then the program will provide a message to the user.

****
* PROGRAM FILE 120 *
****

```
0010 PRINT "PROGRAM FILE 120"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 120.0"
0100 PRINT
0110 PRINT "        NO ARGUMENT WITH THE EXP FUNCTION"
0120 PRINT
0130 PRINT
0140 PRINT "                                BEGIN TEST."
0150 PRINT
0160 LET Y=EXP
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                 END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160
```

121.0 SYNTAX DIAGNOSTIC - NO ARGUMENT LIST WITH INT

This section tests whether the processor will provide a syntax diagnostic when INT has no argument in line 160 (see sections 7 and 8 of BSR X3.60). If the statement is ignored then the program will provide a message to the user.

****
* PROGRAM FILE 121 *
****

```
0010 PRINT "PROGRAM FILE 121"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 121.0"
0100 PRINT
0110 PRINT "          NO ARGUMENT WITH THE INT FUNCTION "
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 LET Y=INT
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                   END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160
```

122.0 SYNTAX DIAGNOSTIC - NO ARGUMENT LIST WITH LOG

This section tests whether the processor will provide a syntax diagnostic when LOG has no argument in line 160 (see sections 7 and 8 of BSR X3.60). If the statement is ignored then the program will provide a message to the user.

****
* PROGRAM FILE 122 *
****

```
0010 PRINT "PROGRAM FILE 122"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 122.0"
0100 PRINT
0110 PRINT "            NO ARGUMENT LIST WITH LOG FUNCTION"
0120 PRINT
0130 PRINT
0140 PRINT "                                BEGIN TEST."
0150 PRINT
0160 LET Y=LOG
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                 END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160
```

123.0 SYNTAX DIAGNOSTIC - NOARGUMENT LIST WITH SGN

This section tests whether the processor will provide a syntax diagnostic when SGN has no argument in line 160 (see section 8 of BSR X3.60). If the statement is ignored then the program will provide a message to the user.

****
* PROGRAM FILE 123 *
****

```
0010 PRINT "PROGRAM FILE 123"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 123.0"
0100 PRINT
0110 PRINT "              NO ARGUMENT LIST WITH THE SGN FUNCTION"
0120 PRINT
0130 PRINT
0140 PRINT "                                BEGIN TEST."
0150 PRINT
0160 LET Y=SGN
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                 END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160
```

124.0 SYNTAX DIAGNOSTIC - NO ARGUMENT LIST WITH SIN

This section tests whether the processor will provide a syntax diagnostic when SIN has no argument in line 160 (see section 8 of BSR X3.60). If the statement is ignored then the program will provide a message to the user.

****
* PROGRAM FILE 124 *
****

```
0010 PRINT "PROGRAM FILE 124"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 124.0"
0100 PRINT
0110 PRINT "             NO ARGUMENT LIST WITH THE SIN FUNCTION"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 LET Y=SIN
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                    END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160
```

125.0 SYNTAX DIAGNOSTIC - NO ARGUMENT LIST WITH SQR

This section tests whether the processor will provide a syntax diagnostic when SQR has no argument in line 160 (section 8 of BSR X3.60). If the statement is ignored then the program will provide a message to the user.

****
* PROGRAM FILE 125 *
****

```
0010 PRINT "PROGRAM FILE 125"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 125.0"
0100 PRINT
0110 PRINT "           NO ARGUMENT LIST WITH THE SQR FUNCTION"
0120 PRINT
0130 PRINT
0140 PRINT "                                BEGIN TEST."
0150 PRINT
0160 LET Y=SQR
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                  END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160
```

126.0 SYNTAX DIAGNOSTIC - NO ARGUMENT LIST WITH TAN

This section tests whether the processor will provide a syntax diagnostic when TAN has no argument in line 160 (see section 8 of BSR X3.60). If the statement is ignored then the program will provide a message to the user.

****
* PROGRAM FILE 126 *
****

```
0010 PRINT "PROGRAM FILE 126"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 126.0"
0100 PRINT
0110 PRINT "            NO ARGUMENT LIST WITH THE TAN FUNCTION"
0120 PRINT
0130 PRINT
0140 PRINT "                                BEGIN TEST."
0150 PRINT
0160 LET Y=TAN
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                 END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160
```

127.0 SYNTAX DIAGNOSTIC - A DEFINED FUNCTION WITH AN ARGUMENT BUT CALLED WITHOUT IT

This section tests whether the processor will provide a syntax diagnostic for a user-defined function called without an argument in line 170 assuming that it had been defined with one in line 160 (see section 8 of BSR X3.60). If the statement is ignored then the program will provide a message to the user.

****
* PROGRAM FILE 127 *
****

```
0010 PRINT "PROGRAM FILE 127"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 127.0"
0100 PRINT
0110 PRINT "   DEFINED FUNCTION WITH A PARAMETER USED WITHOUT IT"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 DEF FNA(X)=X^2-1
0170 LET X=FNA+15
0180 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0190 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0200 PRINT Y
0210 PRINT
0220 PRINT "                                   END TEST."
0230 PRINT
0240 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 170
```

128.0 SYNTAX DIAGNOSTIC - ABS USED WITH TOO MANY ARGUMENTS

This section tests whether a processor will provide a syntax diagnostic for an ABS function used with two arguments in line 210 (see section 8 of BSR X3.60). If this statement is ignored then the program prints a message to the user indicating test failure.

****
* PROGRAM FILE 128 *
****

```
0010 PRINT "PROGRAM FILE 128"
0020 PRINT
0030 PRINT
0040 PRINT
0140 PRINT "                          SECTION 128.0"
0150 PRINT
0160 PRINT "                  ABS WITH TOO MANY ARGUMENTS"
0170 PRINT
0180 PRINT
0190 PRINT "                                  BEGIN TEST."
0200 PRINT
0210 LET Y=ABS(-5,4)
0220 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0230 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0240 PRINT Y
0250 PRINT
0260 PRINT "                                   END TEST."
0270 PRINT
0280 END
```

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

We will indicate two possible diagnostic messages that might be appropriate here. The first would seem more informative.

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 210
```

or

?, WAS SEEN WHERE) WAS EXPECTED IN LINE 210

129.0 SYNTAX DIAGNOSTIC - ATN WITH TOO MANY ARGUMENTS

This section tests whether a processor will provide a syntax diagnostic for an ATN function used with two arguments in line 160 (see section 8 of BSR X3.60). If the statement is ignored then the program prints a message to the user indicating test failure.

****
* PROGRAM FILE 129 *
****

```
0010 PRINT "PROGRAM FILE 129"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 129.0"
0100 PRINT
0110 PRINT "                ATN WITH TOO MANY ARGUMENTS"
0120 PRINT
0130 PRINT
0140 PRINT "                                    BEGIN TEST."
0150 PRINT
0160 LET Y=ATN(-1.57,1.57079)
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                     END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160

or

?, WAS SEEN WHERE ) WAS EXPECTED IN LINE 160
```

130.0 SYNTAX DIAGNOSTIC - COS USED WITH TOO MANY ARGUMENTS

This section tests whether a processor will provide a syntax diagnostic for a COS function used with two arguments in line see section 8 of BSR X3.60). If this statement is ignored then the program prints a message to the user indicating test failure.

****
* PROGRAM FILE 130 *
****

```
0010 PRINT "PROGRAM FILE 130"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 130.0"
0100 PRINT
0110 PRINT "             COS WITH TOO MANY ARGUMENTS"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 LET Y=COS(1.5,1.37)
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                   END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160

or

?, WAS SEEN WHERE ) WAS EXPECTED IN LINE 160
```

131.0 SYNTAX DIAGNOSTIC - EXP USED WITH TOO MANY ARGUMENTS

This section tests whether a processor will provide a syntax diagnostic for an EXP function used with two arguments in line 160 (see section 8 of BSR X3.60). If this statement is ignored then the program prints a message to the user indicating test failure.

****
* PROGRAM FILE 131 *
****

```
0010 PRINT "PROGRAM FILE 131"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 131.0"
0100 PRINT
0110 PRINT "               EXP WITH TOO MANY ARGUMENTS"
0120 PRINT
0130 PRINT
0140 PRINT "                                   BEGIN TEST."
0150 PRINT
0160 LET Y=EXP(5,-2)
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                    END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160

or

?, WAS SEEN WHERE ) WAS EXPECTED IN LINE 160
```

132.0 SYNTAX DIAGNOSTIC - INT USED WIITH TOO MANY ARGUMENTS

This section tests whether a processor will provide a syntax diagnostic for the INT function used with two arguments in line 160 (see section 8 of BSR X3.60). If this statement is ignored then tthe program prints a message to tthe user indicating test failure.

****
* PROGRAM FILE 132 *
****

```
0010 PRINT "PROGRAM FILE 132"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 132.0"
0100 PRINT
0110 PRINT "               INT WITH TOO MANY ARGUMENTS"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 LET Y=INT(1.3,-1.3)
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                   END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160

or

?, WAS SEEN WHERE ) WAS EXPECTED IN LINE 160
```

133.0 SYNTAX DIAGNOSTIC - LOG USED WITH TOO MANY ARGUMENTS

This section tests whether a processor will provide a syntax diagnostic for the LOG function used with two arguments in line 160 (see section 8 of BSR X3.60). If this statement is ignored then the program prints a message to the user indicating test failure.

****
* PROGRAM FILE 133 *
****

```
0010 PRINT "PROGRAM FILE 138"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 133.0"
0100 PRINT
0110 PRINT "                LOG WITH TOO MANY ARGUMENTS"
0120 PRINT
0130 PRINT
0140 PRINT "                                   BEGIN TEST."
0150 PRINT
0160 LET Y=LOG(15,25)
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                    END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160
```

or

```
?, WAS SEEN WHERE ) WAS EXPECTED IN LINE 160
```

134.0 SYNTAX DIAGNOSTIC - SGN USED WITH TOO MANY ARGUMENTS

This section tests whether a processor will provide a syntax diagnostic for a SGN function used with two arguments in line 160 (see section 8 of BSR X3.60). If this statement is ignored then the program prints a message to the user indicating test failure.

****
* PROGRAM FILE 134 *
****

```
0010 PRINT "PROGRAM FILE 134"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 134.0"
0100 PRINT
0110 PRINT "                  SGN WITH TOO MANY ARGUMENTS"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 LET Y=SGN(-15,-18)
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOW THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                   END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160

or

?, WAS SEEN WHERE ) WAS EXPECTED IN LINE 160
```

135.0 SYNTAX DIAGNOSTIC - SIN USED WITH TOO MANY ARGUMENTS

This section tests whether a processor will provide a syntax diagnostic for a SIN function used with two arguments in line 160 (see section 8 of BSR X3.60). If this statement is ignored then the program prints a message to the user indicating test failure.

****
* PROGRAM FILE 135 *
****

```
0010 PRINT "PROGRAM FILE 135"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 135.0"
0100 PRINT
0110 PRINT "                SIN WITH TOO MANY ARGUMENTS"
0120 PRINT
0130 PRINT
0140 PRINT "                                   BEGIN TEST."
0150 PRINT
0160 LET Y=SIN(1.375,1.57)
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                    END TEST."
0220 PRINT
0230 END
```

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

```
?1NCORRECT NUMBER OF ARGUMENTS IN LINE 160

OR

?, WAS SEEN WHERE ) WAS EXPECTED IN LINE 160
```

136.0 SYNTAX DIAGNOSTIC - SQR USED WITH TOO MANY ARGUMENTS

This section tests whether a processor will provide a syntax diagnostic for a SQR function used with two arguments in line 160 (see section 8 of BSR X3.60). If this statement is ignored then the program prints a message to the user indicating test failure.

****
* PROGRAM FILE 136 *
****

```
0010 PRINT "PROGRAM FILE 136"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 136.0"
0100 PRINT
0110 PRINT "               SQR WITH TOO MANY ARGUMENTS"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 LET Y=SQR(16,25)
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                   END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160

or

?, WAS SEEN WHERE ) WAS EXPECTED IN LINE 160
```

137.0 SYNTAX DIAGNOSTIC - TAN USED WITH TOO MANY ARGUMENTS

This section tests whether a processor will provide a syntax diagnostic for a TAN function used with two arguments in line 160 (see section 8 of BSR X3.60). If this statement is ignored then the program prints a message to the user indicating test failure.

****
* PROGRAM FILE 137 *
****

```
0010 PRINT "PROGRAM FILE 137"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 137.0"
0100 PRINT
0110 PRINT "               TAN WITH TOO MANY ARGUMENTS"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 LET Y=TAN(1.5707,-1.5707)
0170 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THE STATE-"
0180 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0190 PRINT Y
0200 PRINT
0210 PRINT "                                   END TEST."
0220 PRINT
0230 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 160

or

?, WAS SEEN WHERE ) WAS EXPECTED IN LINE 160
```

138.0 SYNTAX DIAGNOSTIC - DEFINED FUNCTION WITH MORE THAN ONE ARGUMENT

This section tests whether a processor will provide a syntax diagnostic for a user defined function that is called with too many argumments in line 190 (see section 16 of BSR X3.60). If this statement is ignored then the program prints a message to the user indicating test failure.

****
* PROGRAM FILE 138 *
****

```
0010 PRINT "PROGRAM FILE 138"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 138.0"
0100 PRINT
0110 PRINT "        DEFINED FUNCTION WITH MORE THAN ONE ARGUMENT"
0120 PRINT
0130 PRINT
0140 PRINT "                                  BEGIN TEST."
0150 PRINT
0160 DEF FNA(X)=X*A+B
0170 LET A=5
0180 LET B=6
0190 LET Y=FNA(5,6)
0200 PRINT "     WHETHER OR NOT A NUMERICAL VALUE FOLLOWS THIS STATE-"
0210 PRINT "MENT, THE SYSTEM HAS FAILED THE TEST."
0220 PRINT Y
0230 PRINT
0240 PRINT "                                    END TEST."
0250 PRINT
0260 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 190
```

or

?, WAS SEEN WHERE) WAS EXPECTED IN LINE 190

139.0 SYNTAX DIAGNOSTIC - A DEFINED FUNCTION WITH ILLEGAL ARGUMENT

This section tests whether a processor will provide a syntax diagnostic for a user defined function that is called with an argument in line 130, when the function was defined without an argument in line 120 (see section 16 of BSR X3.60). If the statement is ignored the program will print a message to the user indicating test failure.

```
                        ********************
                        * PROGRAM FILE 139 *
                        ********************
10 PRINT "PROGRAM FILE 139"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "                                SECTION 139.0"
60 PRINT
70 PRINT "                  DEFINED FUNCTION WITH ILLEGAL ARGUMENT"
80 PRINT
90 PRINT
100 PRINT "                                BEGIN TEST"
110 PRINT
120 DEF FNE = 2.7183
130 LET Z = FNE(2)
140 PRINT "THE SYSTEM FAILS THE TEST IF THIS LINE IS PRINTED WITH NO"
145 PRINT "DIAGNOSTIC MESSAGE."
150 PRINT
160 PRINT "                                END TEST"
170 END
```

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

```
?INCORRECT NUMBER OF ARGUMENTS IN LINE 130
```

140.0 SYNTAX DIAGNOSTIC - REFERENCE TO AN UNDEFINED FUNCTION

This section tests whether a processor will provide a syntax diagnostic when a program makes reference to an undefined function in line 170 (see section 16 of BSR X3.60). The user should be warned that some systems might diagnose this syntax error as a reference to an illegal variable. Others may correctly diagnose the undefined function as such. If the system totally ignores the statement then a failure diagnostic is included in the test.

****
* PROGRAM FILE 140 *
****

```
10 PRINT "PROGRAM FILE 140"
20 PRINT
30 PRINT
40 PRINT
90 PRINT "                                SECTION 140.0"
100 PRINT
110 PRINT "                    REFERENCE TO AN UNDEFINED FUNCTION"
120 PRINT
130 PRINT
140 PRINT
150 PRINT "                                BEGIN TEST"
160 PRINT
170 LET Y = FNX(15) + 3
180 PRINT "IF THIS LINE IS PRINTED THE SYSTEM HAS FAILED THE TEST"
200 PRINT
210 PRINT "                                END TEST"
220 END
```

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

```
?UNDEFINED FUNCTION IN LINE 170
```

or

```
?ILLEGAL VARIABLE IN LINE 170
```

141.0 TESTING THE STOP STATEMENT

The objective of this test is to verify that the implementation will recognize the STOP statement in the course of executing a program (see section 10 of BSR X3.60). Program execution must be terminated at that point.

The test is structured to give two possible messages to the user. The first message informs the user that after it, the execution of the test should terminate. The user should consider the implementation to have passed the test, if indeed the program terminated. If the second message also appears, then the implementation has ignored the STOP statement between the two statements, without recognizing the STOP.

If the implementation passes the test, only the following message should be printed:

IF PROGRAM EXECUTION TERMINATES AFTER THIS STATEMENT, THE STOP STATEMENT FOR THIS SYSTEM WORKS.

However, should the implementation fail the test, the following message will also be printed:

PROGRAM EXECUTION WAS NOT TERMINATED, THEREFORE, THE SYSTEM HAS FAILED THE TEST.

There is one point that must be made here. If the system under test does not recognize STOP and does not provide a diagnostic or continue execution by ignoring it, then the only way that a user can determine whether STOP has been successful is by other system-related information being provided after the execution of the program. For example, at the ordinary termination of a program some systems may give execution time and then return to the BASIC processor command mode. If the program halts on STOP without without returning to the processor, then the user might be able to conclude that the STOP statement was illegal in some form. At this point, though, the user should consult with the local system expert to determine that the halt is not caused by some other source.

```
**********************
* PROGRAM FILE 141 *
**********************
```

```
0010 PRINT "PROGRAM FILE 141"
0020 PRINT
0030 PRINT
0040 PRINT
0110 PRINT "                    SECTION 141.0 THE STOP STATEMENT."
0120 PRINT
```

```
0130 PRINT
0140 PRINT "                                   BEGIN TEST."
0150 PRINT
0160 PRINT "     IF PROGRAM EXECUTION TERMINATES AFTER THIS STATEMENT,"
0170 PRINT "THE STOP STATEMENT FOR THIS SYSTEM WORKS."
0180 STOP
0190 PRINT "     PROGRAM EXECUTION WAS NOT TERMINATED, THEREFORE, THE"
0200 PRINT "SYSTEM HAS FAILED THE TEST."
0210 PRINT
0220 PRINT "                                     END TEST."
0230 PRINT
0240 END
```

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

PROGRAM FILE 141

SECTION 141.0 THE STOP STATEMENT.

BEGIN TEST.

IF PROGRAM EXECUTION TERMINATES AFTER THIS STATEMENT,
THE STOP STATEMENT FOR THIS SYSTEM WORKS.

142.0 END STATEMENT MUST BE THE LAST PROGRAM STATEMENT

This section tests the requirement that the END statement be physically last in the program (see section 4 of BSR X3.60). We have introduced an END statement in line 130 and we expect that the system will provide a diagnostic pointing to line 130. This is a reasonable situation to diagnose when one considers that several individual program sections might be written and tested separately and combined into a larger program at a later time. An END statement from one of the modules might have been missed by the programmer after joining the individual programs.

****
* PROGRAM FILE 142 *
****

```
10 PRINT "PROGRAM FILE 142"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "                          SECTION 142.0"
60 PRINT
65 PRINT "               END STATEMENT MUST BE LAST"
70 PRINT
80 PRINT "                          BEGIN TEST"
90 PRINT 1.0
100 GO TO 200
110 PRINT 3.0
120 GO TO 220
125 PRINT "                          END TEST"
130 END
200 PRINT 2.0
210 GO TO 110
220 PRINT 4.0
230 GO TO 125
240 END
```

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

```
?END IS NOT LAST IN LINE 130
```

143.0 A PROGRAM REQUIRES AN END STATEMENT

In this section we will test a simple program without an END statement (see section 4 of BSR X3.60). The system should provide a diagnostic for this situation. If it does not, the program will indicate that the test fails.

```
**********************
* PROGRAM FILE 143 *
**********************
```

```
10 PRINT "PROGRAM FILE 143"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "                              SECTION 143.0"
55 PRINT
60 PRINT "                  PROGRAM REQUIRES AN END STATEMENT"
70 PRINT
80 PRINT "IF THIS STATEMENT APPEARS THE SYSTEM HAS NOT SCANNED AHEAD"
90 PRINT "TO FIND THAT THERE IS NO END STATEMENT. IF THERE IS NO"
100 PRINT "DIAGNOSTIC INDICATING THIS THEN THE TEST HAS FAILED."
```

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

```
?END STATEMENT IS MISSING
```

144.0 COMPOUND EXPRESSIONS AS ARGUMENTS FOR ABS AND ATN

The objective of this test section is to determine whether the implementation will let the argument of a function call be expressed in various numerical expression forms (see sections 7 and 8 of BSR X3.60).

144.1 The ABS Function

This subtest takes an incremented approach in determining the kinds of expressions allowed as the argument-list of a function reference. It begins with expressions involving addition of terms and then advances through other hierachical forms of operations which include the nesting of an ABS function with an ABS function. On output, the test can give three types of output messages to the user. First, should the implementation fail to correctly evaluate any one of the expressions, an error message will be printed indicating which expression failed. Second, there should be an overall error message to the user, if any expression failed to be evaluated properly. Third, should the implementation not fail any of the evaluations, then the message printed should inform the user that the implementation passed the test.

144.2 The ATN Function

This subtest is similar to section 144.1. However, in this case, the ATN function is being used instead of ABS. The approach to testing the correct evaluations of the expressions used as the arguments is the same. On output, the format should follow the same pattern described in section 144.1.

```
**********************
* PROGRAM FILE 144 *
**********************
```

```
0010 PRINT "PROGRAM FILE 144"
0020 PRINT
0030 PRINT
0040 PRINT
0110 PRINT "SECTION 144.0:COMPOUND EXPRESSIONS AS FUNCTION ARGUMENTS"
0130 PRINT
0140 PRINT
0150 PRINT
0160 PRINT "                          SECTION 144.1"
0170 PRINT
0180 PRINT "                           (THE ABS FUNCTION.)"
0190 PRINT
0200 PRINT
0210 PRINT
```

```
0220 PRINT "                                        BEGIN TEST."
0230 PRINT
0240 LET A1=0.5
0250 LET B1=-.25
0260 LET C1=16.0
0270 LET D1=-4.0
0280 LET P=1
0290 LET E1=ABS(A1+B1)+1.0
0300 LET F1=ABS(E1-(0.5-.25)-1.0)
0310 LET E1=ABS(0.0-ABS(A1-C1+D1))
0320 LET G1=ABS(E1+(0.5-16.0-4.0))
0330 LET E1=ABS(A1+1.0-(C1+D1)+0.5*8.0)
0340 LET H1=ABS(E1+(0.5)+1.0-(16.0-4.0)+4.0)
0350 LET E1=ABS(1.0E0+(1.0*1.0/1.0)^2)
0360 LET I1=ABS(E1-2.0)
0370 LET E1=ABS(0-ABS(-5*1/5-5*ABS(-1)))
0380 LET J1=ABS(E1-6)
0390 IF F1<=1E-5 THEN 430
0400 LET R1=1
0410 LET K=1
0420 GOSUB 980
0430 IF G1<=1E-4 THEN 470
0440 LET R1=1
0450 LET K=2
0460 GOSUB 980
0470 IF H1<=1E-5 THEN 510
0480 LET R1=1
0490 LET K=3
0500 GOSUB 980
0510 IF I1<=1E-5 THEN 550
0520 LET R1=1
0530 LET K=4
0540 GOSUB 980
0550 IF J1<=1E-5 THEN 590
0560 LET R1=1
0570 LET K=5
0580 GOSUB 980
0590 LET A$=" ABS "
0600 GOSUB 1080
0610 PRINT
0620 PRINT
0630 PRINT
0640 PRINT "                              SECTION 144.2"
0650 PRINT
0660 PRINT "                               (THE ATN FUNCTION.)"
0670 PRINT
0680 PRINT
0690 PRINT
0700 PRINT "                                    BEGIN TEST."
0710 PRINT
0720 LET A1=.175
0730 LET B1=.5
0740 LET C1=-.6
0750 LET D1=.3
0760 LET P=1
0770 LET E1=ATN(B1^2+2*B1*D1+D1^2)
0780 LET F1=ABS(E1-.569313)
```

```
0790 LET E1=ATN(A1+ABS(C1))
0800 LET G1=ABS(E1-.65931)
0810 LET E1=ATN(B1*ABS(-500))
0820 LET H1=ABS(E1-1.5668)
0830 IF F1<=1E-6 THEN 870
0840 LET R1=2
0850 LET K=1
0860 GOSUB 980
0870 IF G1<=1E-6 THEN 910
0880 LET R1=2
0890 LET K=2
0900 GOSUB 980
0910 IF H1<=1E-5 THEN 950
0920 LET R1=2
0930 LET K=3
0940 GOSUB 980
0950 LET A$=" ATN "
0960 GOSUB 1080
0970 GOTO 1170
0980 ON R1 GOTO 990,1010
0990 LET A$=" ABS "
1000 GOTO 1020
1010 LET A$=" ATN "
1020 PRINT "THE";A$;"FUNCTION FOR THIS SYSTEM HAS FAILED TO EVALUATE"
1030 PRINT "THE COMPOUND EXPRESSION ARGUMENT";K;"THAT WAS REQUIRED OF"
1040 PRINT "IT."
1050 PRINT
1060 LET P=2
1070 RETURN
1080 ON P GOTO 1090,1120
1090 PRINT "THE SYSTEM PASSED THE";A$;"EVALUATION OF COMPOUND EXPRESS-"
1100 PRINT "IONS, AS ITS ARGUMENT."
1110 GOTO 1140
1120 PRINT "THE SYSTEM FAILED THE";A$;"EVALUATION OF COMPOUND EXPRESS-"
1130 PRINT "IONS, AS ITS ARGUMENT, BECAUSE OF THE ABOVE REASONS."
1140 PRINT
1150 PRINT "                                    END TEST."
1160 RETURN
1170 PRINT
1180 END
```

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

PROGRAM FILE 144

SECTION 144.0:COMPOUND EXPRESSIONS AS FUNCTION ARGUMENTS

SECTION 144.1

(THE ABS FUNCTION.)

BEGIN TEST.

THE SYSTEM PASSED THE ABS EVALUATION OF COMPOUND EXPRESSIONS, AS ITS ARGUMENT.

END TEST.

SECTION 144.2

(THE ATN FUNCTION.)

BEGIN TEST.

THE SYSTEM PASSED THE ATN EVALUATION OF COMPOUND EXPRESSIONS, AS ITS ARGUMENT.

END TEST.

145.0 COMPOUND EXPRESSIONS AS ARGUMENTS FOR COS AND EXP

145.1 The COS Function

This test is similar to section 144.1, except for the use of COS. On output, it uses the same formatted messages as described in 144.1.

145.2 The EXP function

This test is again similar to section 144.1 except for the use of EXP. The output messages are also the same.

```
**********
* PROGRAM FILE 145 *
**********
```

```
0010 PRINT "PROGRAM FILE 145"
0020 PRINT
0030 PRINT
0040 PRINT
0050 PRINT "SECTION 145.0:COMPOUND EXPRESSIONS AS ARGUMENTS"
0060 PRINT
0070 PRINT
0080 PRINT
0090 PRINT "                          SECTION 145.1"
0100 PRINT
0110 PRINT "                           (THE COS FUNCTION.)"
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "                                 BEGIN TEST."
0160 PRINT
0170 LET A1=1.0
0180 LET B1=2.0
0190 LET C1=.5
0200 LET P=1
0210 LET D1=COS(B1-2.0*A1)
0220 LET E1=ABS(D1-1.0)
0230 LET D1=COS(A1^2-2*A1*C1+C1^2)
0240 LET F1=ABS(D1-.968912)
0250 LET D1=COS(A1+B1+6*C1)
0260 LET G1=ABS(D1-.960177)
0270 IF E1<=1E-5 THEN 310
0280 LET R1=1
0290 LET K=1
0300 GOSUB 820
0310 IF F1<=1E-6 THEN 350
```

```
0320 LET R1=1
0330 LET K=2
0340 GOSUB 820
0350 IF G1<=1E-6 THEN 390
0360 LET R1=1
0370 LET K=3
0380 GOSUB 820
0390 LET A$=" COS "
0400 GOSUB 920
0410 PRINT
0420 PRINT
0430 PRINT
0440 PRINT "                          SECTION 145.2"
0450 PRINT
0460 PRINT "                           (THE EXP FUNCTION.)"
0470 PRINT
0480 PRINT
0490 PRINT
0500 PRINT "                                 BEGIN TEST."
0510 PRINT
0520 LET A1=-16.0
0530 LET B1=4.0
0540 LET P=1
0550 LET C1=EXP(3*B1+A1)
0560 LET D1=ABS((C1-1.83156E-2)/1.83156E-2)
0570 LET C1=EXP(B1^2-A1/2)
0580 LET E1=ABS((C1-2.64891E10)/2.64891E10)
0590 LET C1=EXP(ABS(A1)+A1)
0600 LET F1=ABS(C1-1.0)
0610 LET C1=EXP(B1+ABS(A1)-4.0)
0620 LET G1=ABS((C1-.888611E7)/.888611E7)
0630 IF D1<=5.45983E-6 THEN 670
0640 LET R1=2
0650 LET K=1
0660 GOSUB 820
0670 IF E1<=3.77514E-6 THEN 710
0680 LET R1=2
0690 LET K=2
0700 GOSUB 820
0710 IF F1<=1E-5 THEN 750
0720 LET R1=2
0730 LET K=3
0740 GOSUB 820
0750 IF G1<=1.12535E-6 THEN 790
0760 LET R1=2
0770 LET K=4
0780 GOSUB 820
0790 LET A$=" EXP "
0800 GOSUB 920
0810 GOTO 1010
0820 ON R1 GOTO 830,850
0830 LET A$=" COS "
0840 GOTO 860
0850 LET A$=" EXP "
0860 PRINT "THE";A$;"FUNCTION FOR THIS SYSTEM HAS FAILED TO EVALUATE"
0870 PRINT "THE COMPOUND EXPRESSION ARGUMENT";K;"THAT WAS REQUIRED OF"
0880 PRINT "IT."
```

```
0890 PRINT
0900 LET P=2
0910 RETURN
0920 ON P GOTO 930,960
0930 PRINT "THE SYSTEM PASSED THE";A$;"EVALUATION OF COMPOUND EXPRESS-"
0940 PRINT "IONS, AS ITS ARGUMENT."
0950 GOTO 980
0960 PRINT "THE SYSTEM FAILED THE";A$;"EVALUATION OF COMPOUND EXPRESS-"
0970 PRINT "IONS, AS ITS ARGUMENT, BECAUSE OF THE ABOVE REASONS."
0980 PRINT
0990 PRINT "                                    END TEST."
1000 RETURN
1010 PRINT
1020 END
```

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

```
PROGRAM FILE 145


SECTION 145.0:COMPOUND EXPRESSIONS AS ARGUMENTS


                        SECTION 145.1

                     (THE COS FUNCTION.)


                           BEGIN TEST.

THE SYSTEM PASSED THE COS EVALUATION OF COMPOUND EXPRESS-
IONS, AS ITS ARGUMENT.

                           END TEST.


                        SECTION 145.2

                     (THE EXP FUNCTION.)


                           BEGIN TEST.
```

THE SYSTEM PASSED THE EXP EVALUATION OF COMPOUND EXPRESS-
IONS, AS ITS ARGUMENT.

END TEST.

146.0 COMPOUND EXPRESSIONS AS ARGUMENTS FOR LOG AND SGN

146.1 The LOG Function

This test is similar to section 144.1 except for the use of the LOG function. The output messages are also the same.

146.2 The SGN Function

This test is similar to section 144.1 except for the use of the SGN function. The output messages are also the same.

```
**********
* PROGRAM FILE 146 *
**********
```

```
0010 PRINT "PROGRAM FILE 146"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                         SECTION 146.1"
0100 PRINT
0110 PRINT "                          (THE LOG FUNCTION.)"
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "                                BEGIN TEST."
0160 PRINT
0170 LET A1=.125
0180 LET B1=5.0
0190 LET P=1
0200 LET C1=LOG(A1/B1)
0210 LET D1=ABS(C1-(-3.68888))
0220 LET C1=LOG(B1^3+4*A1)
0230 LET E1=ABS(C1-4.83231)
0240 LET C1=LOG(EXP(ABS(-16)))
0250 LET F1=ABS(C1-16)
0260 IF D1<=1E-5 THEN 300
0270 LET R1=1
0280 LET K=1
0290 GOSUB 720
0300 IF E1<=1E-5 THEN 340
0310 LET R1=1
0320 LET K=2
0330 GOSUB 720
0340 IF F1<=1E-4 THEN 380
0350 LET R1=1
```

```
0360 LET K=3
0370 GOSUB 720
0380 LET A$=" LOG "
0390 GOSUB 820
0400 PRINT
0410 PRINT
0420 PRINT
0430 PRINT "                          SECTION 146.2"
0440 PRINT
0450 PRINT "                           (THE SGN FUNCTION.)"
0460 PRINT
0470 PRINT
0480 PRINT
0490 PRINT "                                  BEGIN TEST."
0500 PRINT
0510 LET A1=2.5
0520 LET B1=-8.2
0530 LET P=1
0540 LET C1=SGN((B1^2-A1^3)/B1)
0550 LET D1=SGN(ABS(B1)+B1)
0560 LET E1=SGN(B1^2+2*B1*A1+A1^2)
0570 IF C1=-1 THEN 610
0580 LET R1=2
0590 LET K=1
0600 GOSUB 720
0610 IF D1=0 THEN 650
0620 LET R1=2
0630 LET K=2
0640 GOSUB 720
0650 IF E1=1 THEN 690
0660 LET R1=2
0670 LET K=3
0680 GOSUB 720
0690 LET A$=" SGN "
0700 GOSUB 820
0710 GOTO 910
0720 ON R1 GOTO 730,750
0730 LET A$=" LOG "
0740 GOTO 760
0750 LET A$=" SGN "
0760 PRINT "THE";A$;"FUNCTION FOR THIS SYSTEM HAS FAILED TO EVALUATE"
0770 PRINT "THE COMPOUND EXPRESSION ARGUMENT";K;"THAT WAS REQUIRED OF"
0780 PRINT "IT."
0790 PRINT
0800 LET P=2
0810 RETURN
0820 ON P GOTO 830,860
0830 PRINT "THE SYSTEM PASSED THE";A$;"EVALUATION OF COMPOUND EXPRESS-"
0840 PRINT "IONS, AS ITS ARGUMENT."
0850 GOTO 880
0860 PRINT "THE SYSTEM PASSED THE";A$;"EVALUATION OF COMPOUND EXPRESS-"
0870 PRINT "IONS, AS ITS ARGUMENT, BECAUSE OF THE ABOVE REASONS."
0880 PRINT
0890 PRINT "                                    END TEST."
0900 RETURN
0910 PRINT
0920 END
```

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

PROGRAM FILE 146

SECTION 146.1

(THE LOG FUNCTION.)

BEGIN TEST.

THE SYSTEM PASSED THE LOG EVALUATION OF COMPOUND EXPRESSIONS, AS ITS ARGUMENT.

END TEST.

SECTION 146.2

(THE SGN FUNCTION.)

BEGIN TEST.

THE SYSTEM PASSED THE SGN EVALUATION OF COMPOUND EXPRESSIONS, AS ITS ARGUMENT.

END TEST.

147.0 COMPOUND EXPRESSIONS AS ARGUMENTS FOR SIN AND SQR

147.1 The SIN Function

This test is similar to section 144.1 except for the use of the SIN function. The output messages are also the same.

147.2 The SQR Function

This test is similar to section 144.1 except for the use of the SQR function. The output messages are also the same.

```
**********************
* PROGRAM FILE 147 *
**********************
```

```
0010 PRINT "PROGRAM FILE 147"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 147.1"
0100 PRINT
0110 PRINT "                           (THE SIN FUNCTION.)"
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "                                 BEGIN TEST."
0160 PRINT
0170 LET A1=1.0
0180 LET B1=2.0
0190 LET C1=.5
0200 LET P=1
0210 LET D1=SIN(B1-2.0*A1)
0220 LET E1=ABS(D1-0.0)
0230 LET D1=SIN(A1^2-2*A1*C1+C1^2)
0240 LET F1=ABS(D1-.247404)
0250 LET D1=SIN(A1+B1+6*C1)
0260 LET G1=ABS(D1-(-.279415))
0270 IF E1<=1E-6 THEN 310
0280 LET R1=1
0290 LET K=1
0300 GOSUB 770
0310 IF F1<=1E-6 THEN 350
0320 LET R1=1
0330 LET K=2
0340 GOSUB 770
0350 IF G1<=1E-6 THEN 390
```

```
0360 LET R1=1
0370 LET K=3
0380 GOSUB 770
0390 LET A$=" SIN "
0400 GOSUB 870
0410 PRINT
0420 PRINT
0430 PRINT
0440 PRINT "                          SECTION 147.2"
0450 PRINT
0460 PRINT "                           (THE SQR FUNCTION.)"
0470 PRINT
0480 PRINT
0490 PRINT
0500 PRINT "                                  BEGIN TEST."
0510 PRINT
0520 LET A1=64
0530 LET B1=16
0540 LET C1=.25
0550 LET P=1
0560 LET D1=SQR((A1/16)^2+2*A1*B1+(B1-6)^2)
0570 LET E1=ABS(D1-46.5188)
0580 LET D1=SQR(99.5+SQR(ABS(-.25)))
0590 LET F1=ABS(D1-10)
0600 LET D1=SQR((LOG(EXP(A1)))^2)
0610 LET G1=ABS(D1-64)
0620 IF E1<=1E-4 THEN 660
0630 LET R1=2
0640 LET K=1
0650 GOSUB 770
0660 IF F1<=1E-4 THEN 700
0670 LET R1=2
0680 LET K=2
0690 GOSUB 770
0700 IF G1<=1E-4 THEN 740
0710 LET R1=2
0720 LET K=3
0730 GOSUB 770
0740 LET A$=" SQR "
0750 GOSUB 870
0760 GOTO 960
0770 ON R1 GOTO 780,800
0780 LET A$=" SIN "
0790 GOTO 810
0800 LET A$=" SQR "
0810 PRINT "THE";A$;"FUNCTION FOR THIS SYSTEM HAS FAILED TO EVALUATE"
0820 PRINT "THE COMPOUND EXPRESSION ARGUMENT";K;"THAT WAS REQUIRED OF"
0830 PRINT "IT."
0840 PRINT
0850 LET P=2
0860 RETURN
0870 ON P GOTO 880,910
0880 PRINT "THE SYSTEM PASSED THE";A$;"EVALUATION OF COMPOUND EXPRESS-"
0890 PRINT "IONS, AS ITS ARGUMENT."
0900 GOTO 930
0910 PRINT "THE SYSTEM FAILED THE";A$;"EVALUATION OF COMPOUND EXPRESS-"
0920 PRINT "IONS, AS ITS ARGUMENT, BECAUSE OF THE ABOVE REASONS."
```

```
0930 PRINT
0940 PRINT "                                        END TEST."
0950 RETURN
0960 PRINT
0970 END
```

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

```
PROGRAM FILE 147

                    SECTION 147.1

                     (THE SIN FUNCTION.)


                          BEGIN TEST.

THE SYSTEM PASSED THE SIN EVALUATION OF COMPOUND EXPRESS-
IONS, AS ITS ARGUMENT.

                          END TEST.


                    SECTION 147.2

                     (THE SQR FUNCTION.)


                          BEGIN TEST.

THE SYSTEM PASSED THE SQR EVALUATION OF COMPOUND EXPRESS-
IONS, AS ITS ARGUMENT.

                          END TEST.
```

148.0 COMPOUND EXPRESSIONS AS ARGUMENTS FOR TAN AND INT

148.1 The TAN Function

The test is similar to section 144.1 except for the use of the TAN function. The output messages are also the same.

148.2 The INT Function

The test is similar to section 144.1 except for the use of the INT function. The output messages are also the same.

```
**********
* PROGRAM FILE 148 *
**********
```

```
0010 PRINT "PROGRAM FILE 148"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 148.1"
0100 PRINT
0110 PRINT "                           (THE TAN FUNCTION.)"
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "                                 BEGIN TEST."
0160 PRINT
0170 LET A1=1.5
0180 LET B1=.75
0190 LET C1=.25
0200 LET P=1
0210 LET D1=TAN(A1^2-B1)
0220 LET E1=ABS(D1-14.1014)
0230 LET D1=TAN((SIN(C1+5*(B1/3)))^2+(COS(C1+5*(B1/3)))^2)
0240 LET F1=ABS(D1-1.55741)
0250 LET D1=TAN(ATN(.815))
0260 LET G1=ABS(D1-.815)
0270 IF E1<=1E-4 THEN 310
0280 LET R1=1
0290 LET K=1
0300 GOSUB 700
0310 IF F1<=1E-5 THEN 350
0320 LET R1=1
0330 LET K=2
0340 GOSUB 700
0350 IF G1<=1E-6 THEN 390
```

```
0360 LET R1=1
0370 LET K=3
0380 GOSUB 700
0390 LET A$=" TAN "
0400 GOSUB 800
0410 PRINT
0420 PRINT
0430 PRINT
0440 PRINT "                              SECTION 148.2"
0450 PRINT
0460 PRINT "                               (THE INT FUNCTION.)"
0470 PRINT
0480 PRINT
0490 PRINT
0500 PRINT "                                        BEGIN TEST."
0510 PRINT
0520 LET A1=6.25
0530 LET B1=-8.15
0540 LET P=1
0550 LET D1=INT(A1^3+3*A1^2*B1+3*A1*B1^2+B1^3)
0560 LET C1=ABS(D1-(-7))
0570 LET D1=INT(ABS(A1^3+3*A1^2*B1+3*A1*B1^2+B1^3))
0580 LET E1=ABS(D1-6)
0590 IF C1<=1E-5 THEN 630
0600 LET R1=2
0610 LET K=1
0620 GOSUB 700
0630 IF E1<=1E-5 THEN 670
0640 LET R1=2
0650 LET K=2
0660 GOSUB 700
0670 LET A$=" INT "
0680 GOSUB 800
0690 GOTO 890
0700 ON R1 GOTO 710,730
0710 LET A$=" TAN "
0720 GOTO 740
0730 LET A$=" INT "
0740 PRINT "THE";A$;"FUNCTION FOR THIS SYSTEM HAS FAILED TO EVALUATE"
0750 PRINT "THE COMPOUND EXPRESSION ARGUMENT";K;"THAT WAS REQUIRED OF"
0760 PRINT "IT."
0770 PRINT
0780 LET P=2
0790 RETURN
0800 ON P GOTO 810,840
0810 PRINT "THE SYSTEM PASSED THE";A$;"EVALUATION OF COMPOUND EXPRESS-"
0820 PRINT "IONS, AS ITS ARGUMENT."
0830 GOTO 860
0840 PRINT "THE SYSTEM FAILED THE";A$;"EVALUATION OF COMPOUND EXPRESS-"
0850 PRINT "IONS, AS ITS ARGUMENT, BECAUSE OF THE ABOVE REASONS."
0860 PRINT
0870 PRINT "                                        END TEST."
0880 RETURN
0890 PRINT
0900 END
```

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

PROGRAM FILE 148

SECTION 148.1

(THE TAN FUNCTION.)

BEGIN TEST.

THE SYSTEM PASSED THE TAN EVALUATION OF COMPOUND EXPRESSIONS, AS ITS ARGUMENT.

END TEST.

SECTION 148.2

(THE INT FUNCTION.)

BEGIN TEST.

THE SYSTEM PASSED THE INT EVALUATION OF COMPOUND EXPRESSIONS, AS ITS ARGUMENT.

END TEST.

149.0 COMPOUND EXPRESSIONS AS ARGUMENTS FOR A USER DEFINED FUNCTION

The test is similar to section 144.1 except for the use of a user defined function FNA(X) in line 170 (see sections 7 and 16 of BSR X3.60). The output messages are similar to those in section 144.1.

```
**********
* PROGRAM FILE 149 *
**********
```

```
0010 PRINT "PROGRAM FILE 149"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "                          SECTION 149.0"
0100 PRINT
0110 PRINT "                           (THE DEF FUNCTION.)"
0120 PRINT
0130 PRINT
0140 PRINT
0150 PRINT "                                 BEGIN TEST."
0160 PRINT
0170 DEF FNA(X)=X3-1
0180 LET A1=1.6
0190 LET B1=.5
0200 R1=1
0210 LET X=FNA(SIN(A1)+6*B1)
0220 LET X1=ABS((X-62.9795)/62.9795)
0230 LET X=FNA((SIN(B1))2+(COS(B1))2+3)
0240 LET X2=ABS(X-63)
0250 LET X=FNA(ABS(B12-5*A1)+2*(LOG(EXP(B1))))
0260 LET X3=ABS(X-668.922)
0270 IF X1<=1.58782E-6 THEN 310
0280 LET R1=2
0290 LET K=1
0300 GOSUB 400
0310 IF X2<=1E-4 THEN 350
0320 LET R1=2
0330 LET K=2
0340 GOSUB 400
0350 IF X3<=1E-3 THEN 450
0360 LET R1=2
0370 LET K=3
0380 GOSUB 400
0390 GOTO 450
0400 PRINT "THE DEF FUNCTION FOR THIS SYSTEM HAS FAILED TO EVALUATE"
0410 PRINT "THE COMPOUND EXPRESSION ARGUMENT";K;"THAT WAS REQUIRED OF"
0420 PRINT "IT."
```

```
0430 PRINT
0440 RETURN
0450 ON R1 GOTO 460,480
0460 LET A$=" PASSED "
0470 GOTO 490
0480 LET A$=" FAILED "
0490 PRINT "THE SYSTEM";A$;"THE DEF FUNCTION EVALUATION OF COMPOUND"
0500 PRINT "EXPRESSIONS, AS ITS ARGUMENT."
0510 PRINT
0520 PRINT "                                    END TEST."
0530 PRINT
0540 END
```

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

PROGRAM FILE 149

SECTION 149.0

(THE DEF FUNCTION.)

BEGIN TEST.

THE SYSTEM PASSED THE DEF FUNCTION EVALUATION OF COMPOUND
EXPRESSIONS, AS ITS ARGUMENT.

END TEST.

150.0 COMPOUND EXPRESSIONS USED AS PRINT ITEMS AND TAB ARGUMENTS

The objective of this test section is to determine whether the implementation will recognize a numeric expression when it is placed in a print list as one of the separate print items or as the argument of the TAB function (see section 12 of BSR X3.60).

150.1 Expressions As Print Items

This test prints four expressions. These include: (1) simple variables and constants, (2) an intrinsic function evaluated at a compound expression, (3) an expression that includes the sum of a function plus a variable; and (4) a call to a user defined function. The output lies in two columns with the first column being the expected output and the second giving the implementation evaluations of the expressions.

150.2 Expressions As Arguments of TAB Calls

In this test three expressions are used as arguments for a TAB call. In order to obtain correct output the system must round the value of the expression after evaluation and the TAB to that column. If the evaluations are all proper then an A should appear in column 3, under a column index counter, a B should appear on the second row under the column indices in column 6, and finally a C should appear in column 69 on the third row under the indices.

```
**********************
* PROGRAM FILE 150 *
**********************
```

```
0010 PRINT "PROGRAM FILE 150"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "SECTION 150.0: AS THE PRINT ITEMS OF A PRINT-LIST"
0110 PRINT
0120 PRINT
0130 PRINT
0140 PRINT "                          SECTION 150.1"
0150 PRINT
0160 PRINT "                  (AS PRINT ITEMS IN THE DIRECT SENSE.)"
0170 PRINT
0180 PRINT
0190 PRINT
0200 PRINT "                                BEGIN TEST."
0210 PRINT
```

```
0220 PRINT "     TWO COLUMS OF NUMERALS SHOULD APPEAR BELOW THIS PARA-"
0230 PRINT "GRAPH.  IN COLUMN 1 ARE THE EXPECTED VALUES AND IN COLUMN 2"
0240 PRINT "ARE THOSE VALUES WHICH THE PRINT STATEMENT OF THE SYSTEM"
0250 PRINT "EVALUATED."
0260 PRINT
0270 DEF FNA(X)=X^2+1
0280 LET A1=0.5
0290 LET B1=-.25
0300 LET C1=16.0
0310 LET D1=-4.0
0320 PRINT "COLUMN 1","COLUMN 2"
0330 PRINT
0340 PRINT "-.25 ",3*A1+7*B1
0350 PRINT " 6.5 ",ABS(A1+1.0-(C1+D1)+0.5*8.0)
0360 PRINT " 16.4794 ",SIN(A1^2+ABS(B1))+C1
0370 PRINT " 1.54193 ",FNA(B1+A1)+SIN(A1)
0380 PRINT
0390 PRINT "IF THE NUMERICAL VALUES OF EACH PAIR OF NUMERALS, FORMED BY"
0400 PRINT "THE ROWS OF THE RESPECTIVE TWO COLUMN, ARE THE SAME, THEN"
0410 PRINT "CONSIDER THE SYSTEM TO HAVE PASSED THE TEST."
0420 PRINT
0430 PRINT "                                   END TEST."
0440 PRINT
0450 PRINT
0460 PRINT
0470 PRINT "                          SECTION 150.2"
0480 PRINT
0490 PRINT "   (AS PRINT ITEMS IN THE INDIRECT SENSE, ARGUMENTS OF THE"
0500 PRINT "    TAB-CALL."
0510 PRINT
0520 PRINT
0530 PRINT
0540 PRINT "                                   BEGIN TEST."
0550 PRINT
0560 DEF FNB(X)=X^3-8
0570 LET A1=2.75
0580 LET B1=1.5
0590 LET C1=3.1
0600 PRINT "000000000111111111122222222223333333333444444444445";
0610 PRINT "55555555566666666777"
0620 PRINT "12345678901234567890123456789012345678901234567890";
0630 PRINT "123456789012345678901"
0640 PRINT TAB(2*C1-A1);"A"
0650 PRINT TAB(2*ABS(-B1)+A1);"B"
0660 PRINT TAB(FNB(A1+B1)-.15);"C"
0670 PRINT
0680 PRINT "     IF THE ALPHABETS A, B, AND C ARE PRINTED IN COLUMNS 3,"
0690 PRINT "6, AND 69 RESPECTIVELY, OF THE RESPECTIVE LINES OF WHICH"
0700 PRINT "EACH ALPHABET IS PRINTED, THEN CONSIDER THE SYSTEM TO HAVE"
0710 PRINT "PASSED THE TEST."
0720 PRINT
0730 PRINT "                                   END TEST."
0740 PRINT
0750 END
```

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

PROGRAM FILE 150

SECTION 150.0: AS THE PRINT ITEMS 0F A PRINT-LIST

SECTION 150.1

(AS PRINT ITEMS IN THE DIRECT SENSE.)

BEGIN TEST.

TWO COLUMS OF NUMERALS SHOULD APPEAR BELOW THIS PARAGRAPH. IN COLUMN 1 ARE THE EXPECTED VALUES AND IN COLUMN 2 ARE THOSE VALUES WHICH THE PRINT STATEMENT OF THE SYSTEM EVALUATED.

COLUMN 1	COLUMN 2
-.25	-.25
6.5	6.5
16.4794	16.4794
1.54193	1.54193

IF THE NUMERICAL VALUES OF EACH PAIR OF NUMERALS, FORMED BY THE ROWS OF THE RESPECTIVE TWO COLUMN, ARE THE SAME, THEN CONSIDER THE SYSTEM TO HAVE PASSED THE TEST.

END TEST.

SECTION 150.2

(AS PRINT ITEMS IN THE INDIRECT SENSE, ARGUMENTS OF THE TAB-CALL.

BEGIN TEST.

```
000000000111111111122222222223333333333444444444455555555556666666666777
123456789012345678901234567890123456789012345678901234567890123456789012
  A
     B
```

C

IF THE ALPHABETS A, B, AND C ARE PRINTED IN COLUMNS 3, 6, AND 69 RESPECTIVELY, OF THE RESPECTIVE LINES OF WHICH EACH ALPHABET IS PRINTED, THEN CONSIDER THE SYSTEM TO HAVE PASSED THE TEST.

END TEST.

151.0 COMPOUND EXPRESSIONS USED IN IF - THEN STATEMENTS

This test verifies that the implementation will perform comparisons of numerical expressions containing two or more terms (see section 10 of BSR X3.60). The expressions involve function references as well as the multiple terms. On output, should the implementation incorrectly evaluate the comparison, there will be an error message for each incorrect comparison as well as a final summary message concerning the errors. Should the implementation correctly evaluate the expressions there will only appear a message indicating that the implementation passed the comparison test.

```
**********************
* PROGRAM FILE 151 *
**********************
```

```
0010 PRINT "PROGRAM FILE 151"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT "       SECTION 151.0: COMPOUND EXPRESSIONS IN IF-THEN"
0100 PRINT "                      STATEMENTS."
0110 PRINT
0120 PRINT
0130 PRINT
0140 PRINT "                                   BEGIN TEST."
0150 PRINT
0160 DEF FNA(X)=X^2+2*X+1
0170 LET A1=3.5
0180 LET B1=1.625
0190 LET C1=.815
0200 LET D1=-4.5
0210 LET F=1
0220 IF A1+B1<6 THEN 250
0230 LET K=1
0240 GOSUB 410
0250 IF 1>A1-ABS(D1) THEN 280
0260 LET K=2
0270 GOSUB 410
0280 IF INT(SQR(ABS(D1)))=2 THEN 310
0290 LET K=3
0300 GOSUB 410
0310 IF FNA(A1)+5<=ABS(D1)+A1^3 THEN 340
0320 LET K=4
0330 GOSUB 410
0340 IF EXP(A1)+D1>=(A1-.36)^2+D1 THEN 370
0350 LET K=5
0360 GOSUB 410
0370 IF TAN(C1)+A1<>ATN(C1)+A1 THEN 450
```

```
0380 LET K=6
0390 GOSUB 410
0400 GOTO 450
0410 PRINT "THE SYSTEM FAILED RELATIONAL EVALUATION NUMBER";K;"."
0420 LET F=2
0430 PRINT
0440 RETURN
0450 ON F GOTO 460,490
0460 PRINT "THE SYSTEM PASSED THE RELATIONAL EVALUATION OF THE COMPOUND"
0470 PRINT "EXPRESSIONS AS USED IN THE IF-THEN STATEMENTS."
0480 GOTO 510
0490 PRINT "THE SYSTEM FAILED THE RELATIONAL EVALUATION OF THE COMPOUND"
0500 PRINT "EXPRESSIONS AS USED IN THE IF-THEN STATEMENTS."
0510 PRINT
0520 PRINT "                                   END TEST."
0530 PRINT
0540 END
```

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

PROGRAM FILE 151

SECTION 151.0: COMPOUND EXPRESSIONS IN IF-THEN STATEMENTS.

BEGIN TEST.

THE SYSTEM PASSED THE RELATIONAL EVALUATION OF THE COMPOUND EXPRESSIONS AS USED IN THE IF-THEN STATEMENTS.

END TEST.

152.0 COMPOUND EXPRESSIONS USED IN FOR - NEXT STATEMENTS

The objective of this test is to determine whether the implementation will make proper evaluations when numeric expressions are used as the parameters in a FOR - NEXT statement (see section 11 of BSR X3.60). In fact the numeric expressions are used as the initial-value, the limit, and the step-clause. On output, the test prints one of two possible output messages to the user, i.e. either the system passes or it fails.

```
**********************
* PROGRAM FILE 152 *
**********************
```

```
0010 PRINT "PROGRAM FILE 152"
0020 PRINT
0030 PRINT
0040 PRINT
0090 PRINT " SECTION 152.0: COMPOUND EXPRESSIONS USED IN FOR-NEXT"
0100 PRINT "                STATEMENTS."
0110 PRINT
0120 PRINT
0130 PRINT
0140 PRINT "                                   BEGIN TEST."
0150 PRINT
0160 LET N=3
0170 LET M=6
0180 LET N(1)=10
0190 LET M(1)=4
0200 LET O=-1
0210 LET K=0
0220 FOR I0=N TO M
0230 FOR I1=N(1) TO M(1) STEP O
0240 FOR I2=M^2+2*M*M(1)^2 TO (N(1)^3-1)+ABS(O) STEP 20*N(1)/2
0250 FOR I3=ABS(M(1)-M) TO M(1)^2-M^2 STEP 2*SGN(M(1)^2-M^2)
0260 LET K=K+1
0270 NEXT I3
0280 NEXT I2
0290 NEXT I1
0300 NEXT I0
0310 IF K=2688 THEN 360
0320 PRINT "     THE SYSTEM FAILED TO EVALUATE THE COMPOUND EXPRESSIONS"
0330 PRINT "IN A PROPER MANNER FOR THE CORRECT NUMBER OF FOR-NEXT LOOP-"
0340 PRINT "ING."
0350 GOTO 390
0360 PRINT "     THE SYSTEM PASSED THE EVALUATION OF COMPOUND EXPRESS-"
0370 PRINT "IONS BECAUSE THE PROPER NUMBER OF FOR-NEXT LOOPINGS WERE"
0380 PRINT "PERFORMED."
0390 PRINT
```

```
0400 PRINT "                                   END TEST."
0410 PRINT
0420 END
```

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

```
PROGRAM FILE 152


 SECTION 152.0: COMPOUND EXPRESSIONS USED IN FOR-NEXT
               STATEMENTS.


                          BEGIN TEST.

     THE SYSTEM PASSED THE EVALUATION OF COMPOUND EXPRESS-
IONS BECAUSE THE PROPER NUMBER OF FOR-NEXT LOOPINGS WERE
PERFORMED.

                          END TEST.
```

153.0 COMPOUND EXPRESSIONS USED IN SUBSCRIPTS

This section tests whether the implementation will accept compound numeric expressions as subscripts to dimensioned variables (see section 6 of BSR X3.60). In order to access the array variable however the system must first evaluate the expression and then round the resulting value to the nearest integer. For example, in this test there are four expressions that are used as subscripts in program lines 200, 260 and 320. In the first of these lines, numbers were chosen in such a way that there should be no round-off problem. In the second expression, there is a need for the system to round-down and in the final expression the system will have to round-up plus evaluate another expression for a doubly dimensioned array. If any evaluation fails, a message will point to the line in the program that failed to evaluate properly. There is a final message indicating that the system passed or failed.

```
**********************
* PROGRAM FILE 153 *
**********************
```

```
10 PRINT "PROGRAM FILE 153"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "SECTION 153.0:COMPOUND EXPRESSIONS USED IN SUBSCRIPTS"
110 PRINT
120 PRINT
130 PRINT "                        BEGIN TEST"
140 LET B(1) = 16
150 LET B(2) = 10
160 LET C(3,1) = 15
170 LET F = 0
180 LET X(1) = 7
190 LET A = -9
200 LET D = B((X(1) + ABS(A))/16) - 16
210 IF ABS(D) < 1E-6 THEN 240
220 PRINT "EVALUATION ERROR IN PROGRAM LINE 200"
230 LET F = 1
240 LET X(2) = 2.2
250 LET X(3) = 4.4
260 LET D = B(((2*X(2) + X(3))/8.8) - 9*SIN(0) + 1.2) - 10
270 IF ABS(D) < 1E-6 THEN 300
280 PRINT "EVALUATION ERROR IN PROGRAM LINE 260"
290 LET F = 1
300 LET Q3 = 3
310 LET X(4) = 2.7
320 LET D = C(((X(4) - Q3) + 3),(Q3 - 2)) - 15
330 IF ABS(D) < 1E-6 THEN 360
```

```
340 PRINT "EVALUATION ERROR IN PROGRAM LINE 320"
350 LET F = 1
360 IF ABS(F) < 0.5 THEN 390
370 PRINT "COMPOUND EXPRESSIONS AS SUBSCRIPTS: FAILED"
380 GO TO 400
390 PRINT "COMPOUND EXPRESSIONS AS SUBSCRIPTS: PASSED"
400 PRINT
410 PRINT "                          END TEST"
420 END
```

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

```
PROGRAM FILE 153


SECTION 153.0:COMPOUND EXPRESSIONS USED IN SUBSCRIPTS

                    BEGIN TEST
COMPOUND EXPRESSIONS AS SUBSCRIPTS: PASSED

                        END TEST
```

154.0 A COMPOUND EXPRESSION USED IN AN ON - GOTO STATEMENT

In this test, an ON - GOTO expression is evaluated with three possible transfers (see section 10 of BSR X3.60). If all transfers are made correctly, then there will be three messages indicating each correct transfer. If any transfer is made incorrectly, there will be a message indicating failure. The test is passed if all transfers are correct.

```
**********************
* PROGRAM FILE 154 *
**********************
```

```
10 PRINT "PROGRAM FILE 154"
15 PRINT
20 PRINT
25 PRINT
30 PRINT "SECTION 154.0: COMPOUND EXPRESSION USED IN AN ON - GOTO"
35 PRINT "                STATEMENT"
40 PRINT
45 PRINT "                              BEGIN TEST"
50 PRINT
60 LET A = 3
65 LET B = 1
70 LET F = 0
75 FOR L = 0 TO 2
80 ON ((L+SIN(0)*9)+((A/3)-1))+B GOTO 100, 200, 300
85 PRINT "SYSTEM FELL THROUGH ON - GOTO WHEN L = "; L
90 STOP
100 LET F = 1
110 IF L + F > 1.5 THEN 140
120 PRINT "FIRST ON - GOTO TRANSFER CORRECT"
130 GO TO 400
140 PRINT "ON - GOTO TRANSFER "; L+1;" INCORRECT"
150 GO TO 400
200 LET F = 2
210 IF L+F < 2.5 THEN 250
220 IF L+F > 3.5 THEN 250
230 PRINT "SECOND ON - GOTO TRANSFER CORRECT"
240 GO TO 400
250 PRINT "ON - GOTO TRANSFER";L+1;" INCORRECT"
260 GO TO 400
300 LET F = 3
310 IF L+F < 4.5 THEN 350
320 IF L+F > 5.5 THEN 350
330 PRINT "THIRD ON - GOTO TRANSFER CORRECT"
340 GO TO 400
350 PRINT "ON - GOTO TRANSFER";L+1;" INCORRECT"
400 NEXT L
```

```
410 PRINT
420 PRINT "                                   END TEST"
430 END
```

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

PROGRAM FILE 154

SECTION 154.0: COMPOUND EXPRESSION USED IN AN ON - GOTO STATEMENT

BEGIN TEST

FIRST ON - GOTO TRANSFER CORRECT
SECOND ON - GOTO TRANSFER CORRECT
THIRD ON - GOTO TRANSFER CORRECT

END TEST

155.0 COMPOUND EXPRESSIONS USING SUPPLIED FUNCTIONS

In this section we compute several expressions that involve the supplied functions within them (see section 7 of BSR X3.60). The object is to show that the compound expressions are properly parsed, supplied functions evaluated and operations ordered properly. If any expression is evaluated incorrectly a message will be printed indicating to the user the line number of that expression. Otherwise a single message to the user will indicate that the system passed.

```
**********************
* PROGRAM FILE 155 *
**********************
```

```
10 PRINT "PROGRAM FILE 155"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "SECTION 155.0:COMPOUND EXPRESSIONS USING SUPPLIED FUNCTIONS"
60 PRINT
70 PRINT
80 PRINT
90 PRINT "                                BEGIN TEST"
100 PRINT
110 LET C1 = 1.75
120 LET F = 0
130 LET C2 = LOG(EXP(C1)) - 1.75
140 IF ABS(C2) < 1E-6 THEN 170
150 PRINT "INCORRECT EVALUATION IN LINE 130"
160 LET F = 1
170 LET C3 = EXP(LOG(C1)) - 1.75
180 IF ABS(C3) < 1E-6 THEN 210
190 PRINT "INCORRECT EVALUATION IN LINE 170"
200 LET F = 1
210 LET C4 = (SIN(2.0))^2 + (COS(2.0))^2 - 1.0
220 IF ABS(C4) < 1E-6 THEN 250
230 PRINT "INCORRECT EVALUATION IN LINE 210"
240 LET F = 1
250 LET C5 = (1.0/COS(1.2))^2 - ((SIN(1.2)/COS(1.2))^2) - 1.0
260 IF ABS(C5) < 1E-6 THEN 290
270 PRINT "INCORRECT EVALUATION IN LINE 250"
280 LET F = 1
290 LET C6 = SIN(.78) - SQR(1.0 - (COS(0.78))^2)
300 IF ABS(C6) < 1E-6 THEN 330
310 PRINT "INCORRECT EVALUATION IN LINE 290"
320 LET F = 1
330 LET C7 = COS(1.5) - SQR(1.0 - (SIN(1.5))^2)
340 IF ABS(C7) < 1E-6 THEN 370
```

```
350 PRINT "INCORRECT EVALUATION IN LINE 330"
360 LET F = 1
370 LET C8 = SQR((1.0/COS(0.5236))^2 - 1.0) - (SIN(0.5236)/COS(0.5236))
380 IF ABS(C8) < 1E-6 THEN 410
390 PRINT "INCORRECT EVALUATION IN LINE 370"
400 LET F = 1
410 LET C9 = ATN(SIN(0.5)/COS(0.5)) - 0.5
420 IF ABS(C9) < 1E-6 THEN 450
430 PRINT "INCORRECT EVALUATION IN LINE 410"
440 LET F = 1
450 IF ABS(F) < 0.5 THEN 480
460 PRINT "COMPOUND EXPRESSION TEST FAILED"
470 GO TO 490
480 PRINT "COMPOUND EXPRESSION TEST PASSED"
490 PRINT
500 PRINT "                                   END TEST"
510 PRINT
550 END
```

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

```
PROGRAM FILE 155


SECTION 155.0:COMPOUND EXPRESSIONS USING SUPPLIED FUNCTIONS


                              BEGIN TEST

COMPOUND EXPRESSION TEST PASSED

                              END TEST
```

156.0 SEMANTIC ERROR--UPPER BOUND OF ARRAY SET TO ZERO WITH OPTION

If an option base statement specifies that the lower bound for an array subscript is 1 then no dimension-statement in the program may specify an upper bound of zero (see section 15 of BSR X3.60). The test for this specification is straightforward in that an option base specification will be set to 1 and two arrays, both a single and a double dimensioned array, will have upper bounds set to 0. The system being tested should diagnose this as an incompatible dimension specifications, otherwise the test fails.

****
* PROGRAM FILE 156 *
****

```
10 PRINT "PROGRAM FILE 156"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "SECTION 156.0:  SEMANTIC ERROR--UPPER BOUND OF ARRAY"
60 PRINT "                SET TO ZERO WITH OPTION"
70 PRINT
80 PRINT "                                    BEGIN TEST."
90 OPTION BASE 1
100 DIM A(0),B(0,20)
110 PRINT "IF THIS MESSAGE APPEARS PRIOR TO TERMINATION THEN"
120 PRINT "THE SYSTEM HAS FAILED TO DIAGNOSE AN INCOMPATIBILITY"
130 PRINT "BETWEEN THE OPTION AND DIM STATEMENTS.  THE TEST SYSTEM"
140 PRINT "FAILS."
150 PRINT
160 PRINT "                                      END TEST."
170 END
```

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

```
?DIMENSIONED ARRAYS OUT OF BOUNDS IN LINE 100
```

157.0 SEMANTIC ERROR--MULTIPLE OPTION STATEMENTS

A program may contain at most one option-statement (see section 15.4 of BSR X3.60). An arbitrary program may fail to satisfy this requirement if, for example, it had been created from two separate programs each with option specification molded together as one program. The test system must have the capability of diagnosing that multiple option statements appear in a program. This error is included in the test program below and a diagnostic, otherwise the test system fails.

****
* PROGRAM FILE 157 *
****

```
10 PRINT "PROGRAM FILE 157"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "SECTION 157.0:  SEMANTIC ERROR--MULTIPLE OPTION STATEMENTS"
60 PRINT
70 PRINT "                                   BEGIN TEST."
80 OPTION BASE 1
90 DIM A(30)
100 OPTION BASE 0
110 DIM B(1,20)
120 PRINT "IF THIS STATEMENT APPEARS THEN THE TEST SYSTEM FAILED"
130 PRINT "TO DIAGNOSE TWO OPTION STATEMENTS IN LINES 80 AND 100."
140 PRINT
150 PRINT "                                    END TEST."
160 END
```

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

```
?MULTIPLE OPTION STATEMENT AT LINE 100
```

158.0 SEMANTIC ERROR--OPTION STATEMENT AFTER ARRAY REFERENCE

If an option-statement is present in a program then it must occur in a lower numbered line than any dimension-statement or reference to an array (see section 15.4 of BSR X3.60). This test is a straightforward case analysis in which both implicitly and an explicitly dimensioned arrays appear before an option-statement. The language processor should provide a diagnostic and terminate execution in order to pass this test.

****
* PROGRAM FILE 158 *
****

```
10 PRINT "PROGRAM FILE 158"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "SECTION 158.0:  SEMANTIC ERROR--OPTION STATEMENT AFTER"
60 PRINT "                ARRAY REFERENCE"
70 PRINT
80 PRINT "                                    BEGIN TEST."
90 DIM A(20),B(5,20)
100 LET A(5)=1
110 LET B(4,15)=A(5)+9
120 LET C(10)=5
130 LET D(5,5)=C(10)+B(4,15)-15
140 PRINT D(5,5)
145 GOTO 160
150 OPTION BASE 1
160 PRINT
170 PRINT "IF A ZERO APPEARS ABOVE THIS LINE THE SYSTEM FAILED TO"
180 PRINT "DIAGNOSE THAT THE OPTION-STATEMENT WAS OUT OF ORDER."
190 PRINT
200 PRINT "                                     END TEST."
210 END
```

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

```
?ILLEGAL PLACEMENT OF OPTION IN LINE 150
```

159.0 SEMANTIC ERROR--ARRAY DECLARATION OUT OF PLACE

If a specific declaration by a dimension statement is made for an array then this declaration must appear in a lower numbered line that any reference to an element of that array (see section 15.4 of BSR X3.60). The test for this requirement is straightforward in that arrays are used prior to their dimensioning statement is written. In order to pass, the test system must produce a diagnostic that identifies this error.

```
**********************
* PROGRAM FILE 159 *
**********************
```

```
10 PRINT "PROGRAM FILE 159"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "SECTION 159.0: SEMANTIC ERROR--ARRAY DECLARATION OUT OF PLACE"
60 PRINT
70 PRINT "                                   BEGIN TEST."
80 PRINT
90 LET A(5)=1
100 LET B(5,5)=A(5)+9
110 PRINT B(5,5)
120 GOTO 140
130 DIMENSION A(30),B(10,20)
140 PRINT "IF THIS STATEMENT APPEARS PRECEDED BY A 10 THEN"
150 PRINT "THE SYSTEM FAILED TO RECOGNIZE THE DIMENSION STATEMENT"
160 PRINT "OUT OF PLACE."
170 PRINT
180 PRINT "                                    END TEST."
190 END
```

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

```
?DIMENSION STATEMENT OUT OF PLACE LINE 130
```

160.0 SEMANTIC ERROR--DIMENSIONING AN ARRAY MORE THAN ONCE

If an array is dimensioned explicitly in a program then it can be dimensioned only once (see section 15.4 of BSR X3.60). The test for this is straightforward. An array will be dimensioned twice. This semantic error requires a diagnostic from the test system in order to pass.

****
* PROGRAM FILE 160 *
****

```
10 PRINT "PROGRAM FILE 160"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "SECTION 160.0:  SEMANTIC ERROR--DIMENSIONING AN ARRAY MORE"
60 PRINT "                THAN ONCE"
70 PRINT
80 PRINT "                                   BEGIN TEST."
90 DIM A(25),B(30,10)
100 DIM C(10,20),A(30)
110 PRINT "IF THIS MESSAGE APPEARS THE TEST SYSTEM FAILED TO"
120 PRINT "RECOGNIZE DOUBLE DIMENSIONING IN LINES 80 AND"
130 PRINT "90."
140 PRINT
150 PRINT "                                    END TEST."
160 END
```

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

```
?DIMENSION ERROR LINE 90
```

161.0 INITIALIZING STRING VARIABLES

For string variables there are three commonly used alternatives for associating implementation-defined initial values with string variables. First, an unknown or arbitrary value moght be assigned. Second, the null string might be assigned. Third, all string variables might be detectably undefined in the sense that an exception will result from an attempt to access the variable before that variable is explicitly assigned a value. The American National Standard for Minimal BASIC recommends the third alternative.

In the test below three unassigned string variables are printed. If the values printed are all null then the system probably implements the second alternative. If the values are somewhat arbitrary strings then alternative one has been implemented. In either of these cases the test system fails the standard recommendation. If the test system identifies the lines involving undefined string variables and then terminates it satisfies the reocmmendation.

```
**********************
* PROGRAM FILE 161 *
**********************
```

```
10 PRINT "PROGRAM FILE 161"
20 PRINT
30 PRINT
40 PRINT
50 PRINT "SECTION 161.0:  INITIALIZING STRING VARIABLES"
60 PRINT
70 PRINT "                                    BEGIN TEST."
80 PRINT
90 PRINT "THREE STRING VARIABLES ARE PRINTED BELOW AFTER"
100 PRINT "EACH = SIGN"
110 PRINT
120 PRINT "A$= ";A$
130 PRINT "B$= ";B$
140 PRINT "C$= ";C$
150 PRINT
160 PRINT "IF ALL STRINGS ARE NULL THEN SYSTEM PROBABLY INITIALIZES"
170 PRINT "STRING VARIABLES AS NULL STRINGS."
180 PRINT "IF SOMEWHAT ARBITRARY STRINGS ARE PRINTED THEN THE SYSTEM"
190 PRINT "PROBABLY INITIALIZES STRING VARIABLES RANDOMLY."
200 PRINT "IN EITHER CASE THE SYSTEM FAILS THE STANDARD"
210 PRINT "RECOMMENDATION."
220 PRINT
230 PRINT "                                  END TEST."
240 END
```

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

```
?UNINITIALIZED VARIABLE IN LINE 120
?UNINITIALIZED VARIABLE IN LINE 130
?UNINITIALIZED VARIABLE IN LINE 140
```

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBSIR 78-1420-4	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

NBS Minimal BASIC Test Programs - Version 1
User's Manual
Vol. 4 - Mathematical and User Defined Functions, Compound Expressions

5. Publication Date: January 1978

6. Performing Organization Code

7. AUTHOR(S)
David E. Gilsinn, 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.: 640 1121

11. Contract/Grant No.

12. Sponsoring Organization Name and Complete Address (*Street, City, State, ZIP*)
National Bureau of Standards
Department of Commerce
Washington, DC 20234

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 fourth of four volumes that comprise the user's guide to the NBS Minimal BASIC test programs. They 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 programs in this volume include: mathematical function tests; exception tests for some of the functions; a statistical test of the uniformness of the random number generator; user defined function tests; syntax tests on the functions; tests using compound arithmetic expressions and certain semantic tests on arrays and variable initialization. The entire set of test programs is available on magnetic 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
☒ 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): UNCLASSIFIED

21. NO. OF PAGES: 160

20. SECURITY CLASS (THIS PAGE): UNCLASSIFIED

22. Price

USCOMM-DC 29042-P74