# Test Case for System Test

**System test description:**

Retain all functionalities including output from lab4 while making the below change.

* Create an Identifier Class inherit from Token and modify the Binary Tree to only hold Identifiers.
* Then get rid of LiteralType as a type and use inheritance from a Literal Class instead. We will inherit StringLiteral, IntegerLiteral and RealLiteral classes from the Literal class.

**System test table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function to be test** | **Procedure name** | **Input parameter** | **Output or Return value** | **Expected result** | **Test command line** |
| Extract the content of an input file (NEWTON.PAS) and produce an identical output (ActualOutput.txt) with line numbers then produce a Cross Refrence Information section | int main(int argc, const char \* argv[]) | NEWTON.PAS | MyOutput.txt | Expected output (MyOutput.txt) is identical to input (Sample\_Output.txt) with each line numbered. | ./CrossReference NEWTON.PAS > MyOutput.txt |

# Input data (NEWTON.PAS):

PROGRAM newton (input, output);

CONST

epsilon = 1e-6;

VAR

number, root, sqroot : real;

BEGIN

REPEAT

writeln;

write('Enter new number (0 to quit): ');

read(number);

IF number = 0 THEN BEGIN

writeln(number:12:6, 0.0:12:6);

END

ELSE IF number < 0 THEN BEGIN

writeln('\*\*\* ERROR: number < 0');

END

ELSE BEGIN

sqroot := sqrt(number);

writeln(number:12:6, sqroot:12:6);

writeln;

root := 1;

REPEAT

root := (number/root + root)/2;

writeln(root:24:6,

100\*abs(root - sqroot)/sqroot:12:2,

'%')

UNTIL abs(number/sqr(root) - 1) < epsilon;

END

UNTIL number = 0

END.

# Expected Output data (sample\_output.txt):

\_Page 1 NEWTON.PAS Sat Apr 26 20:07:00 2014

1: PROGRAM newton (input, output);

\_Page 1 NEWTON.PAS Sat Apr 26 20:07:00 2014

>> PROGRAM program

>> <IDENTIFIER> newton

>> ( (

>> <IDENTIFIER> input

>> , ,

>> <IDENTIFIER> output

>> ) )

>> ; ;

2:

3: CONST

>> CONST const

4: epsilon = 1e-6;

>> <IDENTIFIER> epsilon

>> = =

>> <NUMBER> 1e-006 (real)

>> ; ;

5:

6: VAR

>> VAR var

7: number, root, sqroot : real;

>> <IDENTIFIER> number

>> , ,

>> <IDENTIFIER> root

>> , ,

>> <IDENTIFIER> sqroot

>> : :

>> <IDENTIFIER> real

>> ; ;

8:

9: BEGIN

>> BEGIN begin

10: REPEAT

>> REPEAT repeat

11: writeln;

>> <IDENTIFIER> writeln

>> ; ;

12: write('Enter new number (0 to quit): ');

>> <IDENTIFIER> write

>> ( (

>> <STRING> 'Enter new number (0 to quit): '

>> ) )

>> ; ;

13: read(number);

>> <IDENTIFIER> read

>> ( (

>> <IDENTIFIER> number

>> ) )

>> ; ;

14:

15: IF number = 0 THEN BEGIN

>> IF if

>> <IDENTIFIER> number

>> = =

>> <NUMBER> 0 (integer)

>> THEN then

>> BEGIN begin

16: writeln(number:12:6, 0.0:12:6);

>> <IDENTIFIER> writeln

>> ( (

>> <IDENTIFIER> number

>> : :

>> <NUMBER> 12 (integer)

>> : :

>> <NUMBER> 6 (integer)

>> , ,

\_Page 2 NEWTON.PAS Sat Apr 26 20:07:00 2014

>> <NUMBER> 0 (real)

>> <NUMBER> 12 (integer)

>> : :

>> <NUMBER> 6 (integer)

>> ) )

>> ; ;

17: END

>> END end

18: ELSE IF number < 0 THEN BEGIN

>> ELSE else

>> IF if

>> <IDENTIFIER> number

>> < <

>> <NUMBER> 0 (integer)

>> THEN then

>> BEGIN begin

19: writeln('\*\*\* ERROR: number < 0');

>> <IDENTIFIER> writeln

>> ( (

>> <STRING> '\*\*\* ERROR: number < 0'

>> ) )

>> ; ;

20: END

>> END end

21: ELSE BEGIN

>> ELSE else

>> BEGIN begin

22: sqroot := sqrt(number);

>> <IDENTIFIER> sqroot

>> := :=

>> <IDENTIFIER> sqrt

>> ( (

>> <IDENTIFIER> number

>> ) )

>> ; ;

23: writeln(number:12:6, sqroot:12:6);

>> <IDENTIFIER> writeln

>> ( (

>> <IDENTIFIER> number

>> : :

>> <NUMBER> 12 (integer)

>> : :

>> <NUMBER> 6 (integer)

>> , ,

>> <IDENTIFIER> sqroot

>> : :

>> <NUMBER> 12 (integer)

>> : :

>> <NUMBER> 6 (integer)

>> ) )

>> ; ;

24: writeln;

>> <IDENTIFIER> writeln

>> ; ;

25:

26: root := 1;

>> <IDENTIFIER> root

>> := :=

>> <NUMBER> 1 (integer)

>> ; ;

27: REPEAT

\_Page 3 NEWTON.PAS Sat Apr 26 20:07:00 2014

>> REPEAT repeat

28: root := (number/root + root)/2;

>> <IDENTIFIER> root

>> := :=

>> ( (

>> <IDENTIFIER> number

>> / /

>> <IDENTIFIER> root

>> + +

>> <IDENTIFIER> root

>> ) )

>> / /

>> <NUMBER> 2 (integer)

>> ; ;

29: writeln(root:24:6,

>> <IDENTIFIER> writeln

>> ( (

>> <IDENTIFIER> root

>> : :

>> <NUMBER> 24 (integer)

>> : :

>> <NUMBER> 6 (integer)

>> , ,

30: 100\*abs(root - sqroot)/sqroot:12:2,

>> <NUMBER> 100 (integer)

>> \* \*

>> <IDENTIFIER> abs

>> ( (

>> <IDENTIFIER> root

>> - -

>> <IDENTIFIER> sqroot

>> ) )

>> / /

>> <IDENTIFIER> sqroot

>> : :

>> <NUMBER> 12 (integer)

>> : :

>> <NUMBER> 2 (integer)

>> , ,

31: '%')

>> <STRING> '%'

>> ) )

32: UNTIL abs(number/sqr(root) - 1) < epsilon;

>> UNTIL until

>> <IDENTIFIER> abs

>> ( (

>> <IDENTIFIER> number

>> / /

>> <IDENTIFIER> sqr

>> ( (

>> <IDENTIFIER> root

>> ) )

>> - -

>> <NUMBER> 1 (integer)

>> ) )

\_Page 4 NEWTON.PAS Sat Apr 26 20:07:00 2014

>> < <

>> <IDENTIFIER> epsilon

>> ; ;

33: END

>> END end

34: UNTIL number = 0

>> UNTIL until

>> <IDENTIFIER> number

>> = =

>> <NUMBER> 0 (integer)

35: END.

>> END end

>> . .

Cross Reference Information

Identifier Line Numbers

----------- ------------

abs 30 32

epsilon 4 32

input 1

newton 1

number 7 13 15 16 18 22 23 28 32 34

output 1

read 13

real 7

root 7 26 28 28 28 29 30 32

sqr 32

sqroot 7 22 23 30 30

sqrt 22

write 12

writeln 11 16 19 23 24 29