Section 1

Program

```
0
  x = read()
1
  y = read()
2
  z = x + y
3
  # end
```

Write the structural abstraction of the program.

0 expression assignment (+1)

1 expression assignment (+1)

2 expression assignment (+1)

3 #end (+1)

(+1) indentation

(+1) locations

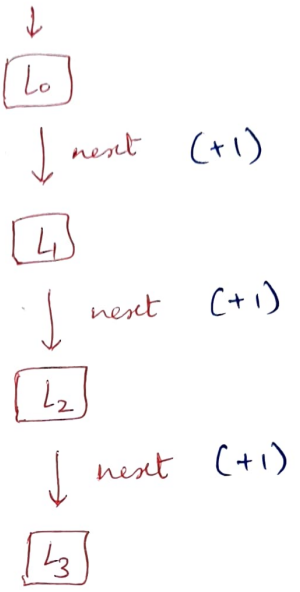
Total : 6

Fill in the table of Control Transfer Functions with the appropriate locations for the program.

i	next	true	false	call	return	error
0	1					3
1	2					3
2	3					3
3						

Total : 6

Draw the Control Flow Graph (CFG) for the program. To reduce clutter, don't draw the Error edges.



Total : 3

Trace the structurally feasible executions of the program.

1. $L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{next} L_3$ (+1)

2. $L_0 \xrightarrow{next} L_1 \xrightarrow{error} L_3$

   All other error executions. } (+1)

TOTAL : 2

Trace the logically feasible executions of the program.

1. x and y are numbers: (+1)

$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{next} L_3$ (+1)

2. Either x or y, or both, are not numbers: (+1)

$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{error} L_3$ (+1)

TOTAL: 4

Trace the actual execution of the program for the inputs x = 2 and y = 3.

$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{next} L_3$

(+1) (+1) (+1)

TOTAL: 3

Section 2

Program

```
0
  g = read()
1
  if g <= 5:
2
      g = g - 4
3
      h = 6 // g
4
  else:
5
      h = g * g
6
  j = g - h
7
  # end
```

Write the structural abstraction of the program.

0 expression assignment (+1)
1 if: (+1)
2 expression assignment (+1)
3 expression assignment (+1)
4 else: (+1)
5 expression assignment (+1)
6 expression assignment (+1)
7 #end (+1)

(+1) indentation
(+1) locations

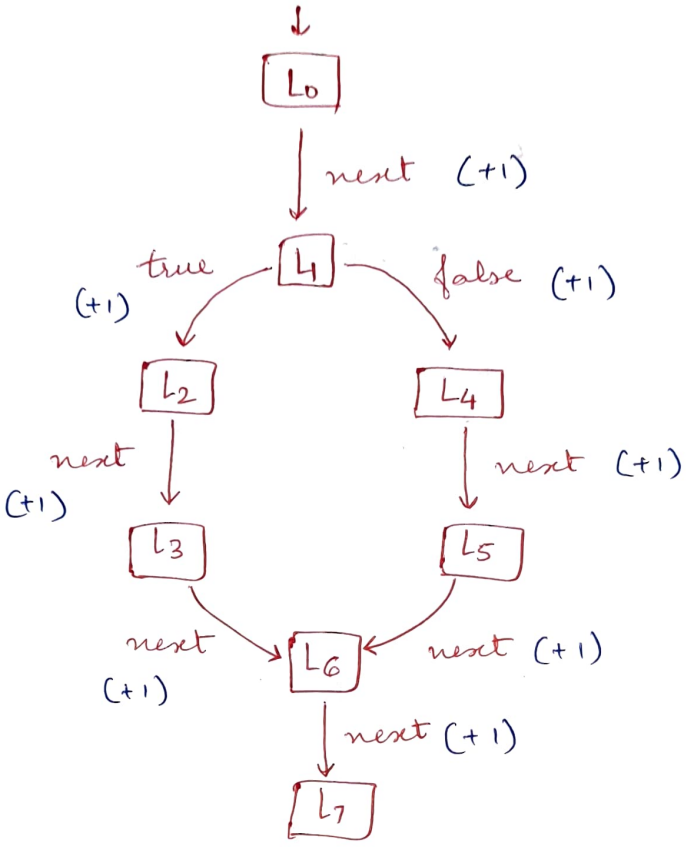
TOTAL: 10

Fill in the table of Control Transfer Functions with the appropriate locations for the program.

i	next	true	false	call	return	error
0	1					7
1		2	4			7
2	3					7
3	6					7
4	5					
5	6					7
6	7					7
7						

TOTAL: 14

Draw the Control Flow Graph (CFG) for the program. To reduce clutter, don't draw the Error edges.



TOTAL : 8

Trace the structurally feasible executions of the program.

1. $L_0 \xrightarrow{next} L_1 \xrightarrow{true} L_2 \xrightarrow{next} L_3 \xrightarrow{next} L_6 \xrightarrow{next} L_7$ (+1)

2. $L_0 \xrightarrow{next} L_1 \xrightarrow{false} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_6 \xrightarrow{next} L_7$ (+1)

3. $L_0 \xrightarrow{next} L_1 \xrightarrow{error} L_7$ } (+1)

All other error executions

TOTAL : 3

Trace the logically feasible executions of the program.

1. $g \leq 5$, and $g \neq 4$ (+1)

$L_0 \xrightarrow{next} L_4 \xrightarrow{true} L_2 \xrightarrow{next} L_3 \xrightarrow{next} L_6 \xrightarrow{next} L_7$ (+1)

2. $g > 5$: (+1)

$L_0 \xrightarrow{next} L_4 \xrightarrow{false} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_6 \xrightarrow{next} L_7$ (+1)

3. g is not a number: (+1)

$L_0 \xrightarrow{next} L_1 \xrightarrow{error} L_7$ (+1)

4. $g = 4$ (+1)

$L_0 \xrightarrow{next} L_1 \xrightarrow{true} L_2 \xrightarrow{next} L_3 \xrightarrow{error} L_7$ (+1)

TOTAL: 8

Trace the actual execution of the program for the input g = 4.

$L_0 \xrightarrow[(+1)]{next} L_1 \xrightarrow[(+1)]{true} L_2 \xrightarrow[(+1)]{next} L_3 \xrightarrow[(+1)]{error} L_7$

TOTAL: 4

Section 3

Program

```
0
  b = read()
1
  x = 5
2
  while b > 0:
3
      b = b - 1
4
      y = x * b
5
      continue
6
  y = b + x
7
  # end
```

Write the structural abstraction of the program.

0 expression assignment (+1)

1 expression assignment (+1)

2 while : (+1)

3 expression assignment (+1)

4 expression assignment (+1)

5 continue (+1)

6 expression assignment (+1)

7 #end (+1)

(+1) indentation

(+1) locations

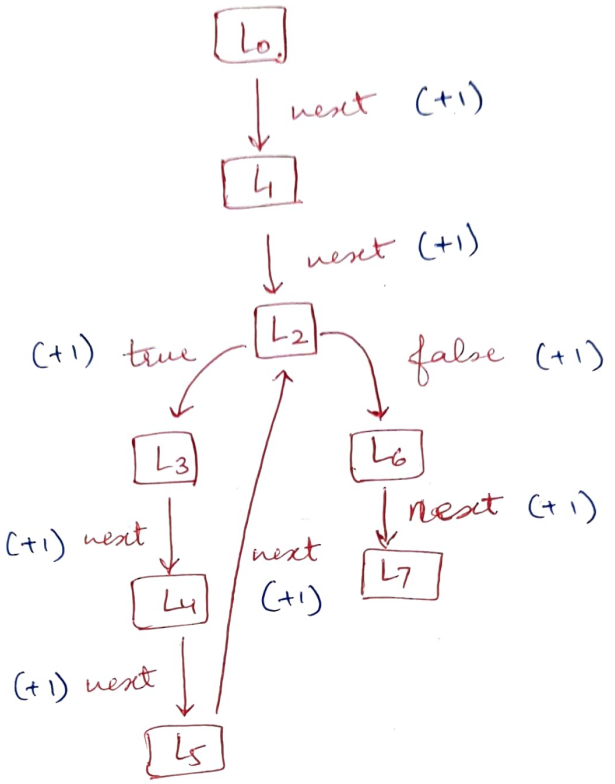
TOTAL: 10

Fill in the table of Control Transfer Functions with the appropriate locations for the program.

i	next	true	false	call	return	error
0	1					7
1	2					7
2		3	6			7
3	4					7
4	5					7
5	2					
6	7					7
7						

TOTAL : 14

Draw the Control Flow Graph (CFG) for the program. To reduce clutter, don't draw the Error edges.



TOTAL : 8

Trace the structurally feasible executions of the program. For executions which enter the while loop, showing one iteration is enough.

1. $L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{true} L_3 \xrightarrow{next} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_2 \xrightarrow{false} L_6 \xrightarrow{next} L_7$ (+1)

2. Executions with multiple such iterations of the body block. (+1)

NOTE: Any indication that the student is aware of the possibility of multiple iterations would be enough to award 1 mark.

3. $L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{false} L_6 \xrightarrow{next} L_7$ (+1)

4. $L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{error} L_7$ } (+1)

All other error executions }

NOTE: Award the mark if they've traced AT LEAST one VALID error execution. The English note alone doesn't carry any mark.

TOTAL : 4

Trace the logically feasible executions of the program. For executions which enter the while loop, showing one iteration is enough.

1. $b > 0$: (+1)

$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{true} L_3 \xrightarrow{next} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_2 \xrightarrow{false} L_6 \xrightarrow{next} L_7$ (+1)

Executions with multiple iterations of the body block based on the value of b. (+1)

2. $b \leq 0$: (+1)

$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{false} L_6 \xrightarrow{next} L_7$ (+1)

3. b is not a number : (+1)

$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{error} L_7$ (+1)

TOTAL : 7

Trace the actual execution of the program for the input b = 2.

$L_0 \xrightarrow[(+1)]{next} L_1 \xrightarrow[(+1)]{next} L_2 \xrightarrow[(+1)]{true} L_3 \xrightarrow[(+1)]{next} L_4 \xrightarrow[(+1)]{next} L_5 \xrightarrow[(+1)]{next} L_2$

$L_2 \xrightarrow[(+1)]{true} L_3 \xrightarrow[(+1)]{next} L_4 \xrightarrow[(+1)]{next} L_5 \xrightarrow[(+1)]{next} L_2 \xrightarrow[(+1)]{false} L_6 \xrightarrow[(+1)]{next} L_7$

TOTAL : 12

Section 4

Program

```
0
  def sub(a, b):
1
    return a - b
2
  def func(x):
3
    x = x * 2
4
    k = sub(x, 1)
5
    return k
6
  p = read()
7
  n = func(p)
8
  z = sub(n, p)
9
  # end
```

Write the structural abstraction of the program.

0 def subs: (+1)
1 return (+1)
2 def func: (+1)
3 expression assignment (+1)
4 call sub assignment (+1)
5 return (+1)
6 expression assignment (+1)
7 call func assignment (+1)
8 call sub assignment (+1)
9 # end (+1)

(+1) indentation
(+1) locations

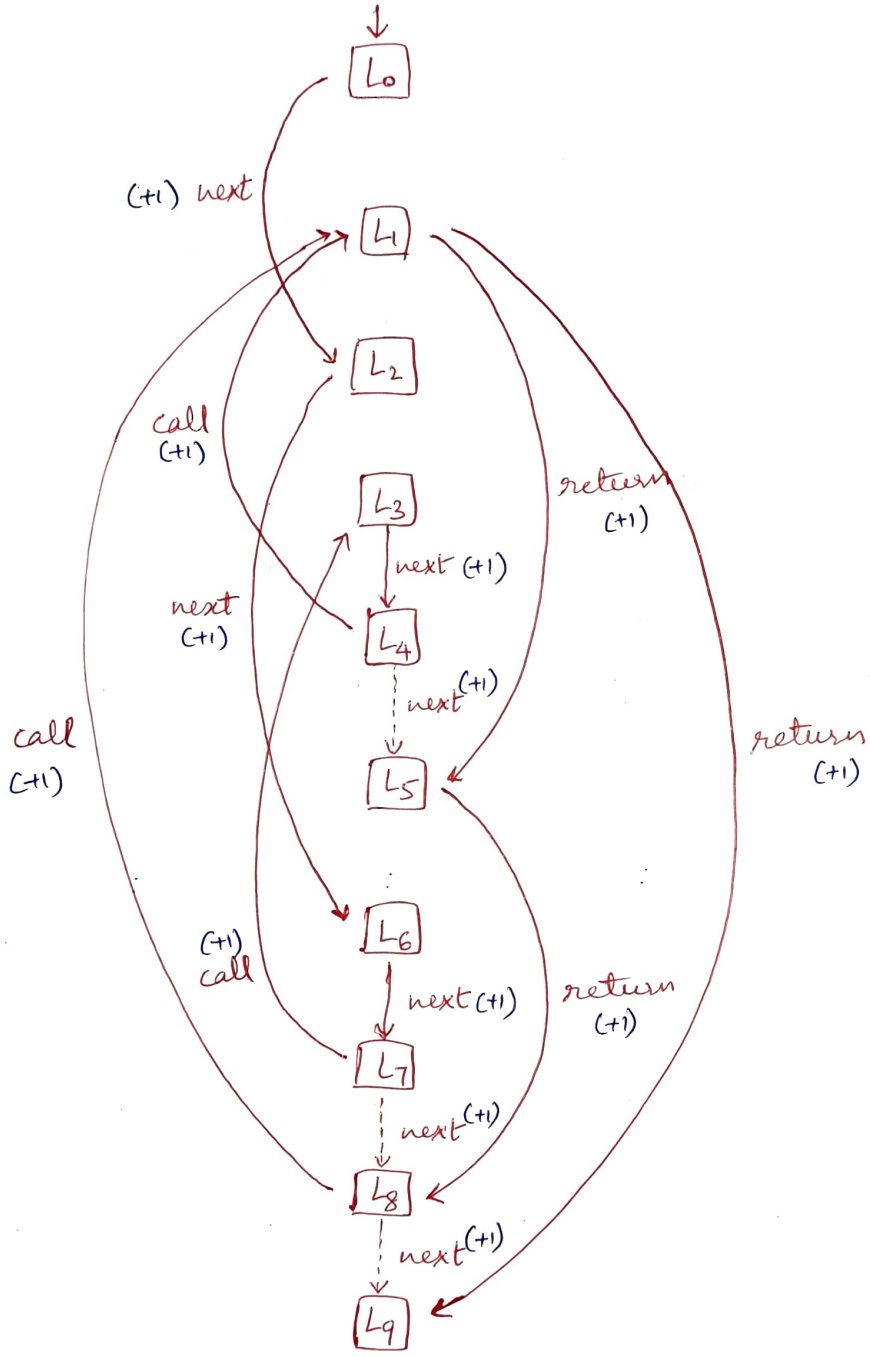
TOTAL: 12

Fill in the table of Control Transfer Functions with the appropriate locations for the program.

i	next	true	false	call	return	error
0	2					
1					{5,9}	7
2	6					
3	4					7
4	5			1		7
5					{8}	7
6	7					7
7	8			3		7
8	9			1		7
9						

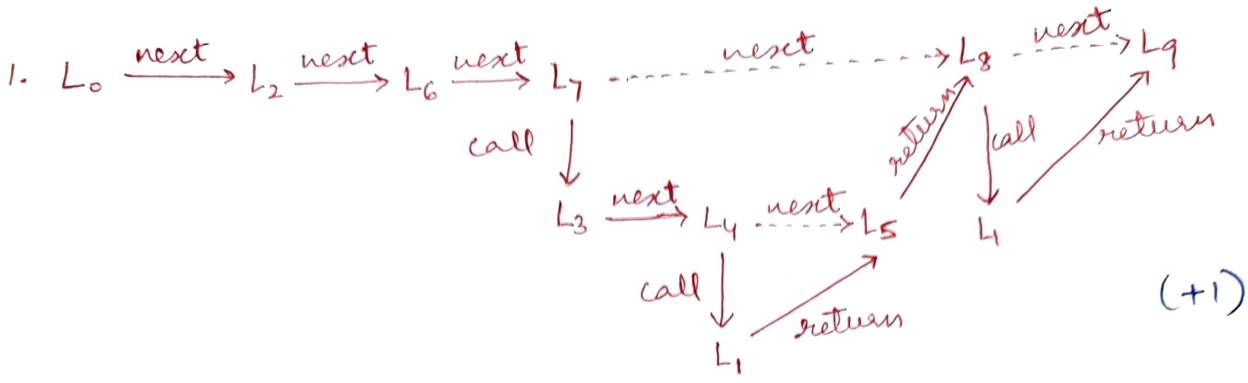
TOTAL: 12

Draw the Control Flow Graph (CFG) for the program. To reduce clutter, don't draw the Error edges.



TOTAL: 13

Trace the structurally feasible executions of the program.

1. 

(+1)

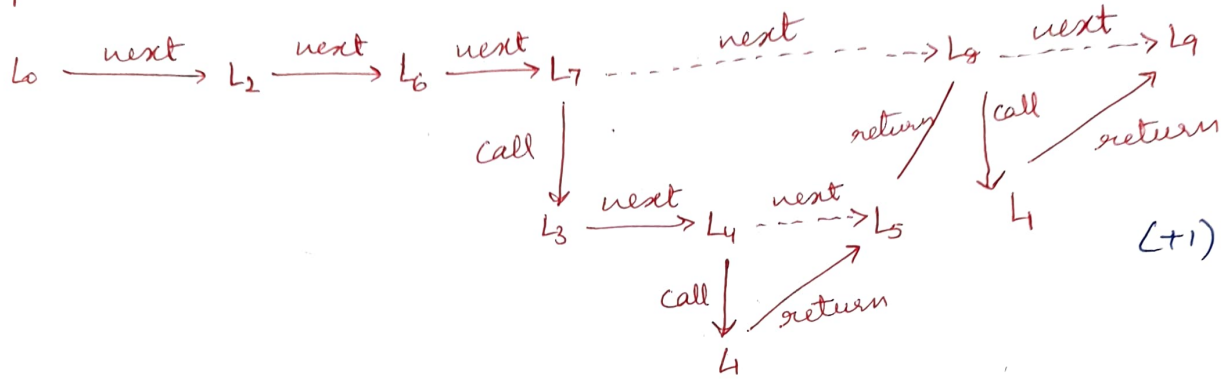
2. L_0 next L_2 next L_6 error L_9 } (+1)

All other error executions

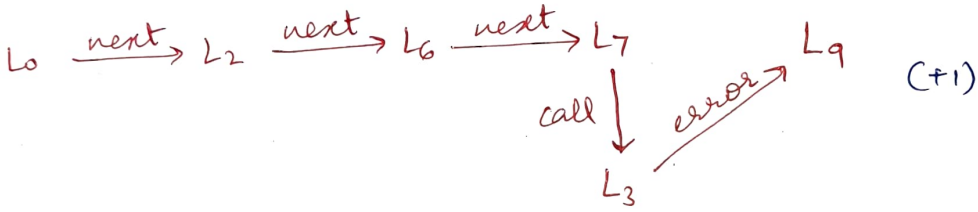
TOTAL : 2

Trace the logically feasible executions of the program.

1. p is a number: (+1)

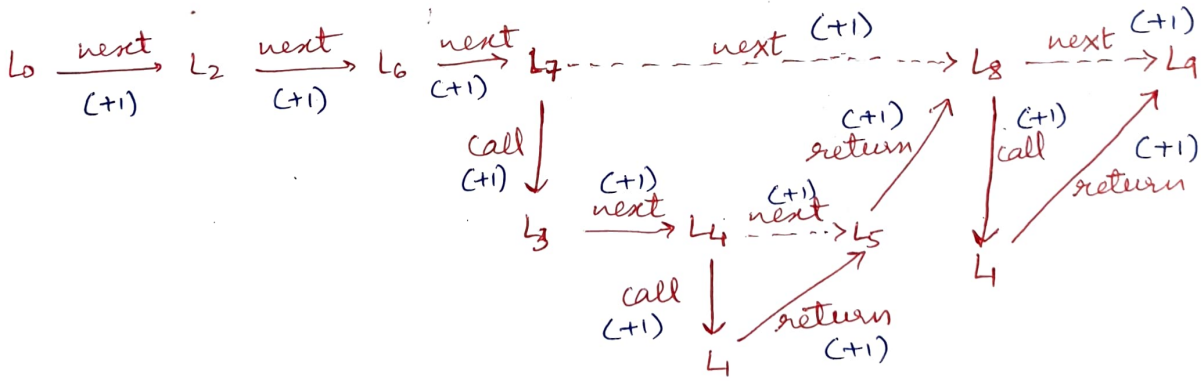


2. p is not a number: (+1)



TOTAL: 4

Trace the actual execution of the program for the input p = 3.



TOTAL: 13

Section 5

Program

```
0
def func(x):
1
    y = 6 // x
2
    return x - 1
3
c = read()
4
while c > 0:
5
    c = func(c)
6
    continue
7
# end
```

TOTAL : 10

Write the structural abstraction of the program.

0 def func: (+1)
1 expression assignment (+1)
2 return (+1)
3 expression assignment (+1)
4 while: (+1)
5 call func assignment (+1)
6 continue (+1)
7 #end (+1)

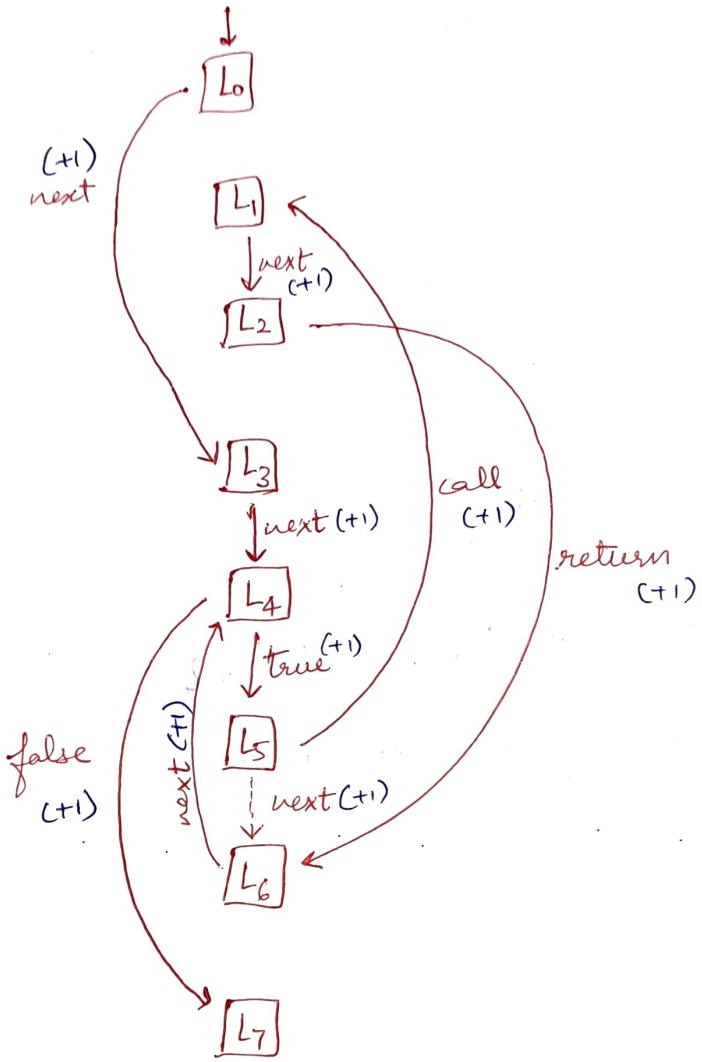
(+1) indentation
(+1) locations

Fill in the table of Control Transfer Functions with the appropriate locations for the program.

i	next	true	false	call	return	error
0	3					
1	2					7
2					{6}	7
3	4					7
4		5	7			7
5	6			1		7
6	4					
7						

TOTAL : 9

Draw the Control Flow Graph (CFG) for the program. To reduce clutter, don't draw the Error edges.



TOTAL : 9

Trace the structurally feasible executions of the program. For executions which enter the while loop, showing one iteration is enough.

1. $L_0 \xrightarrow{next} L_3 \xrightarrow{next} L_4 \xrightarrow{true} L_5 \xrightarrow{call} L_4 \xrightarrow{next} L_2$; $L_6 \xrightarrow{return}$; $L_6 \xrightarrow{next} L_4 \xrightarrow{false} L_7$ (+1)

Executions with multiple such iterations of the body block. (+1)

2. $L_0 \xrightarrow{next} L_3 \xrightarrow{next} L_4 \xrightarrow{false} L_7$ (+1)

3. $L_0 \xrightarrow{next} L_3 \xrightarrow{next} L_4 \xrightarrow{error} L_7$ (+1)

All other error executions

TOTAL : 4

Trace the logically feasible executions of the program. For executions which enter the while loop, showing one iteration is enough.

1. $c > 0$: (+1)

$L_0 \xrightarrow{next} L_3 \xrightarrow{next} L_4 \xrightarrow{true} L_5 \xrightarrow{next} L_6 \xrightarrow{next} L_4 \xrightarrow{false} L_7$ (+1)

$L_5 \xrightarrow{call} L_1 \xrightarrow{next} L_2 \xrightarrow{return} L_6$

Executions with multiple such iterations of body block. (+1)

2. $c \leq 0$: (+1)

$L_0 \xrightarrow{next} L_3 \xrightarrow{next} L_4 \xrightarrow{false} L_7$ (+1)

3. c is not a number: (+1)

$L_0 \xrightarrow{next} L_3 \xrightarrow{next} L_4 \xrightarrow{error} L_7$ (+1)

TOTAL: 7

Trace the actual execution of the program for the input c = 2.

$L_0 \xrightarrow[(+1)]{next} L_3 \xrightarrow[(+1)]{next} L_4 \xrightarrow[(+1)]{true} L_5 \xrightarrow[(+1)]{next} L_6 \xrightarrow[(+1)]{next} L_4 \xrightarrow[(+1)]{true} L_5 \xrightarrow[(+1)]{next} L_6 \xrightarrow[(+1)]{next} L_4 \xrightarrow[(+1)]{false} L_7$

$L_5 \xrightarrow[(+1)]{call} L_1 \xrightarrow[(+1)]{next} L_2 \xrightarrow[(+1)]{return} L_6$

$L_5 \xrightarrow[(+1)]{call} L_1 \xrightarrow[(+1)]{next} L_2 \xrightarrow[(+1)]{return} L_6$

TOTAL: 15