Section 1

Program

```
0
  q = 5
1
  p = read()
2
  m = p + q
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

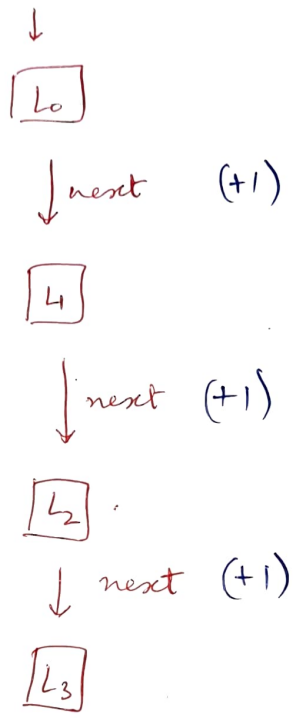
TOTAL: 6

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. p is a number: (+1)

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

2. p is not a number: (+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 input p = 5.

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

TOTAL : 3

Section 2

Program

```
0
  a = 3
1
  b = read()
2
  if b > 5:
3
      a = b // a
4
  else:
5
      a = b
6
  c = a + b
7
  # end
```

Write the structural abstraction of the program.

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

(+1) indentation
(+1) locations

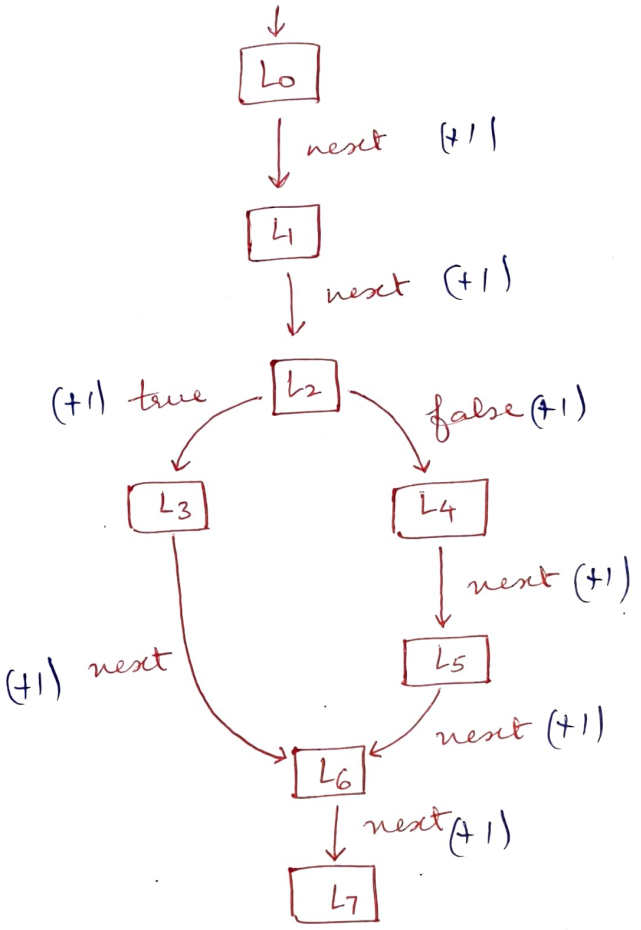
TOTAL : 10

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					7
2		3	4			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. $L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{true} L_3 \xrightarrow{next} L_6 \xrightarrow{next} L_7$ (+1)

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

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

All other error executions.

TOTAL: 3

Trace the logically feasible executions of the program.

1. $b > 5$: (+1)

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

2. $b \leq 5$: (+1)

$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{false} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_6 \xrightarrow{next} L_7$

(+1)

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

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

TOTAL: 5

Trace the actual execution of the program for the input `b = 6`.

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

TOTAL: 5

Section 3

Program

```
0
  p = read()
1
  c = p - 1
2
  while c != 0:
3
      y = p * c
4
      c = c - 1
5
      continue
6
  p = p + 1
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

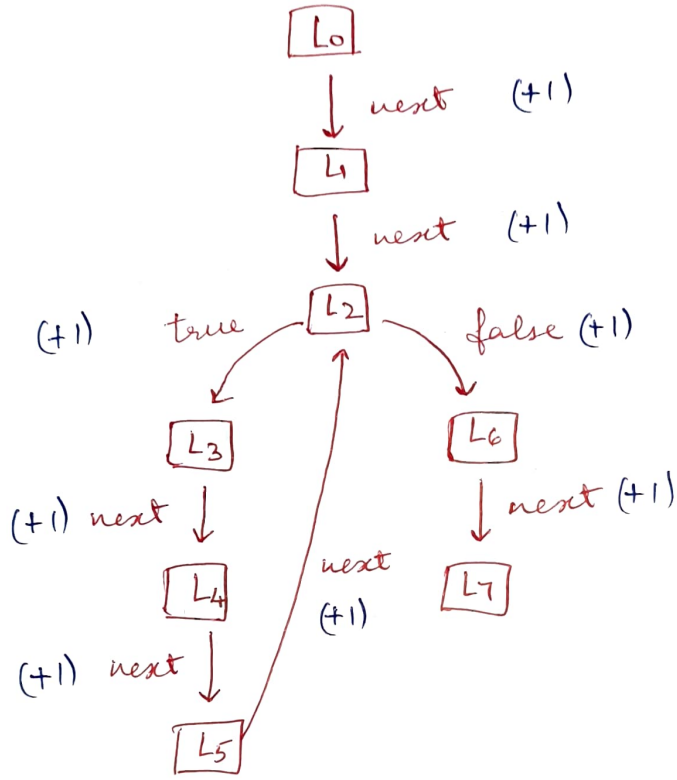
TOTAL: 10

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 loody block. (+1)

NOTE: Any indication that the student is aware of the possibility of multiple iterations ~~xxxxxx~~ 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. $L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{error} L_7$

All other error executions.

} (+1)

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. p = 1 (or) c = 0 : (+1)

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

2. p != 1 (or) c != 0; (+1)

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

Multiple iterations possible including infinite ~~recursion~~ iteration (+1)

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

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

TOTAL : 7

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

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

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

TOTAL : 12

Section 4

Program

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

(+1) locations
(+1) indentation

TOTAL: 12

Write the structural abstraction of the program.

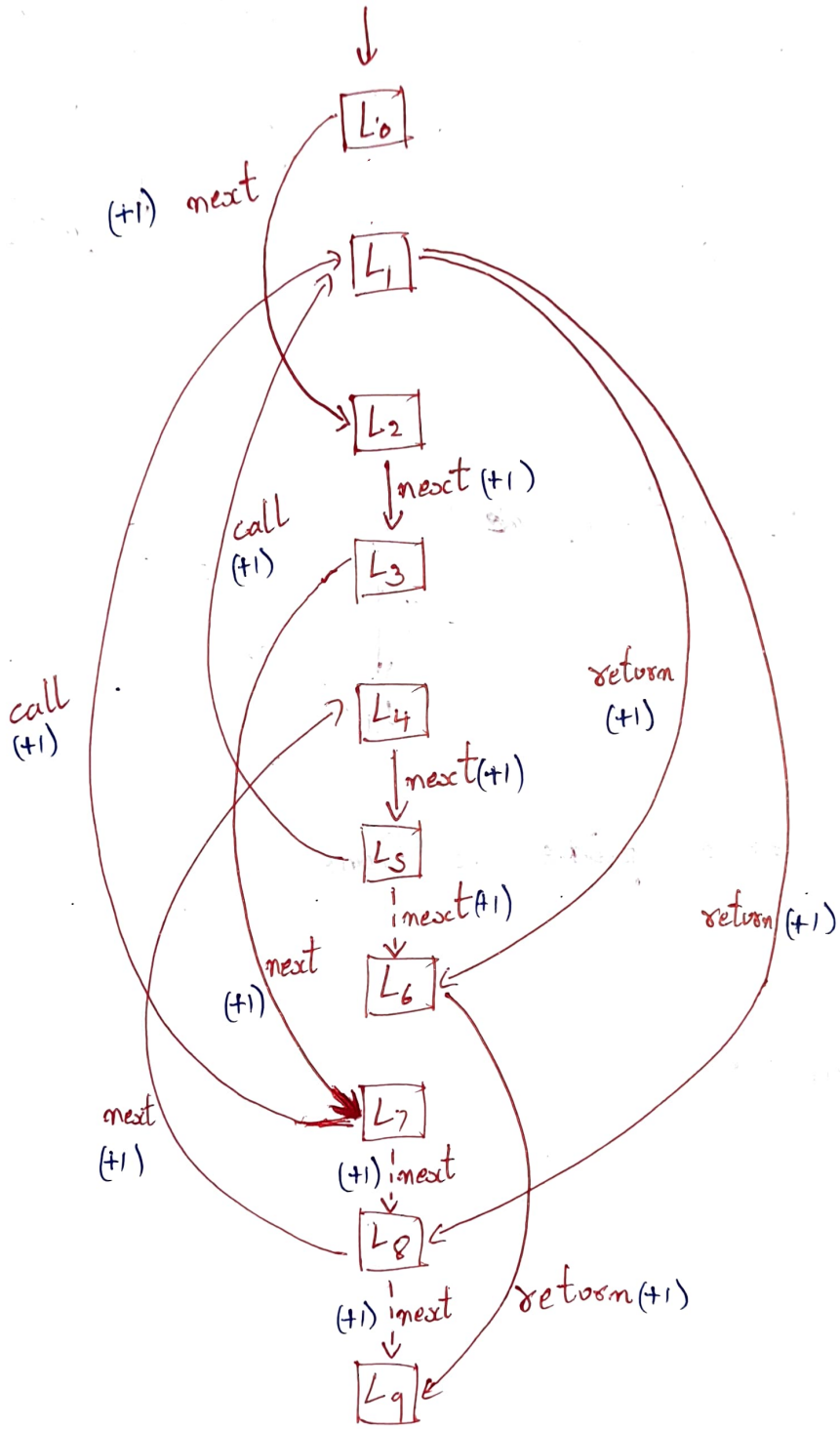
```
0 def div:                          (+1)
1     return                        (+1)
2 expression assignment             (+1)
3 def func:                         (+1)
4     expression assignment         (+1)
5     call div assignment           (+1)
6     return                        (+1)
7 call div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

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, 8}	9
2	3					9
3	7					
4	5					9
5	6			1		9
6					{9}	9
7	8			1		9
8	9			4		9
9						

TOTAL : 19

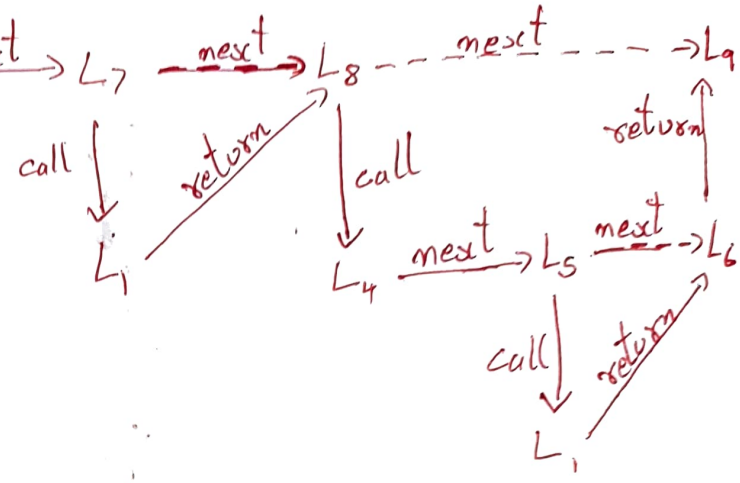
Draw the Control Flow Graph (CFG) for the program. To reduce clutter, don't draw the Error edges.



TOTAL: 13

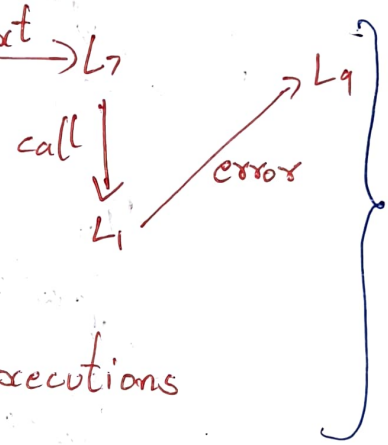
Trace the structurally feasible executions of the program.

1. $L_0 \xrightarrow{next} L_2 \xrightarrow{next} L_3 \xrightarrow{next} L_7 \xrightarrow{next} L_8 \dashrightarrow L_9$



(+1)

2. $L_0 \xrightarrow{next} L_2 \xrightarrow{next} L_3 \xrightarrow{next} L_7$



(+1)

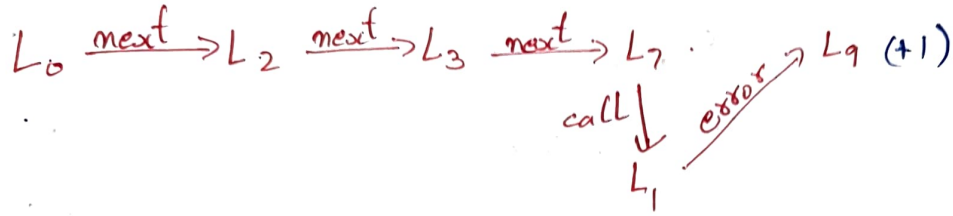
All other error ~~paths~~ executions

TOTAL : 2

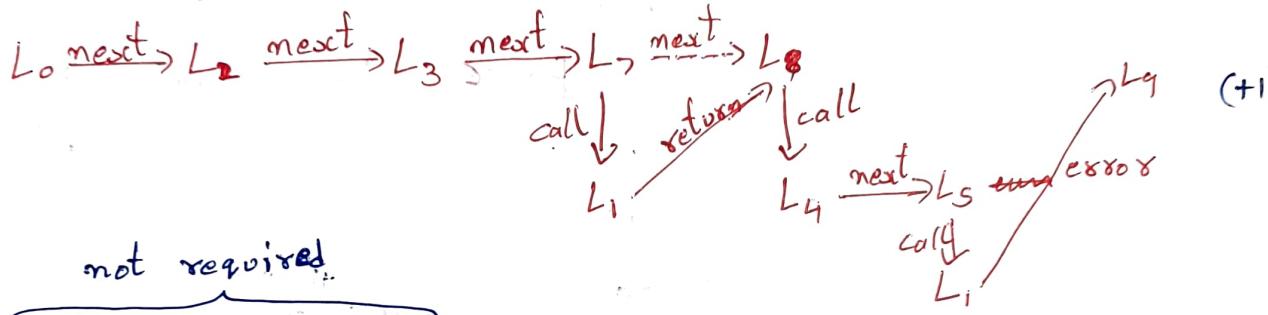
Trace the logically feasible executions of the program.

NOTE: +1 IF BOTH ARE MENTIONED
+1/2 IF ONLY ONE OF THEM IS MENTIONED

1. $x = 0$ or x is not a number: (+1)

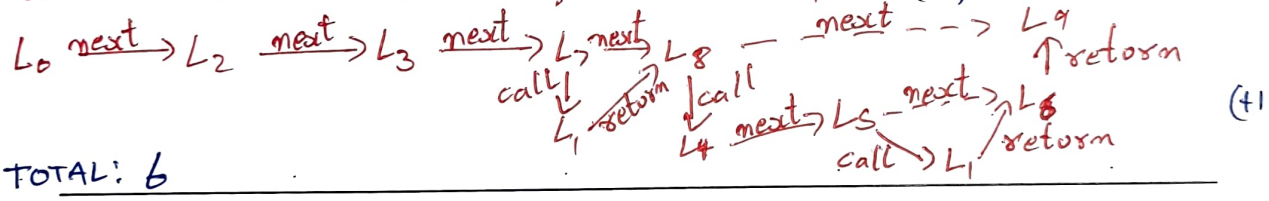


2. $x = 2$: (+1)



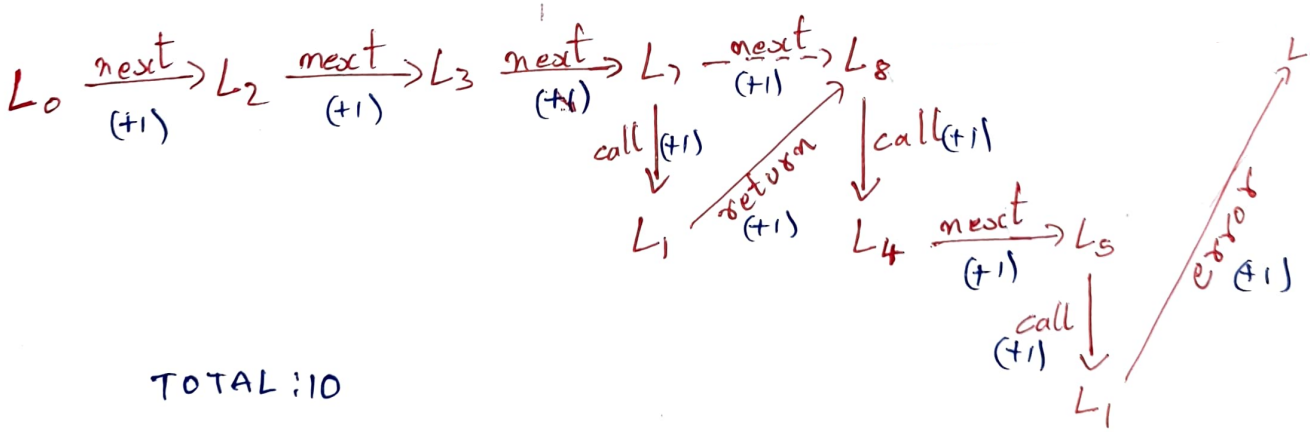
not required

3. x is a number and $x != 0$ and $x != 2$ (+1)



TOTAL: 6

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



TOTAL: 10

Section 5

Program

```
0
def func(y):
1
    return y + 1
2
p = read()
3
while p < 2:
4
    p = func(p)
5
    continue
6
p = p - 1
7
# end
```

Write the structural abstraction of the program.

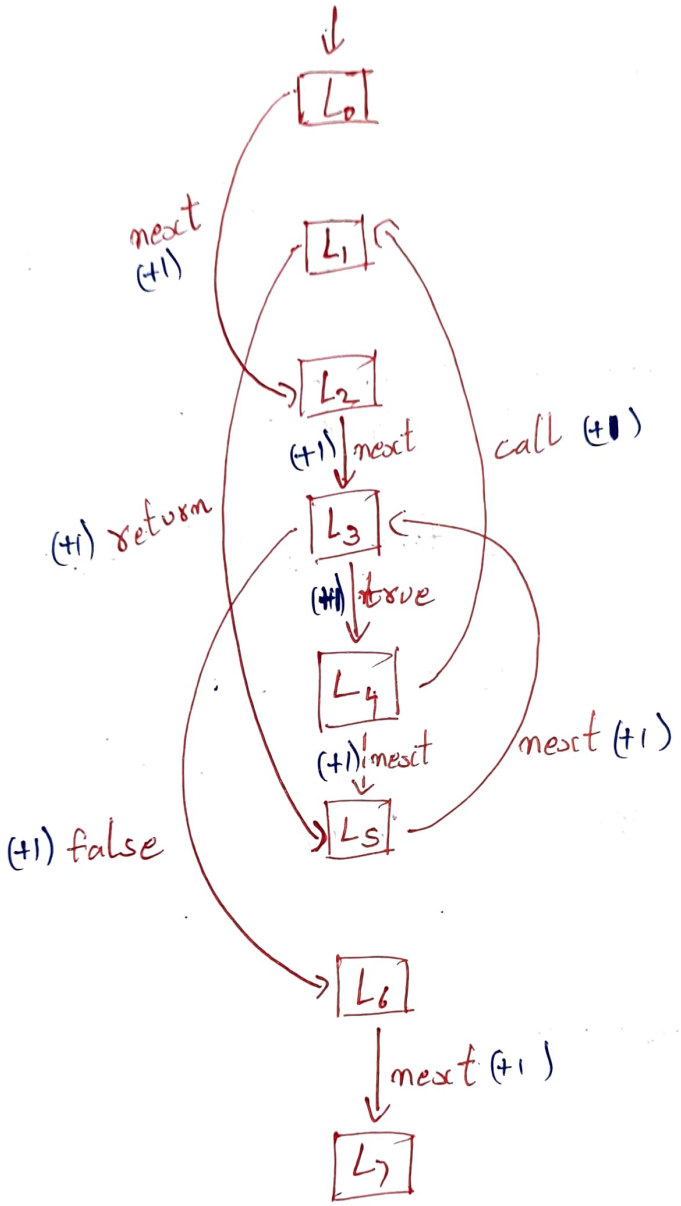
0 def func: (+1)
1 return (+1)
2 expression assignment (+1)
3 while; (+1)
4 call func assignment (+1)
5 continue (+1)
6 expression assignment (+1)
7 #end (+1) locations (+1)
TOTAL: 10 (+1) indentation

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}	7
2	3					7
3		4	6			7
4	5			1		7
5	3					
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: 9

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

1. $L_0 \xrightarrow{next} L_2 \xrightarrow{next} L_3 \xrightarrow{true} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_3$

   $L_4 \xrightarrow{call} L_1 \xrightarrow{return} L_5$ (+1)

   $L_3 \xrightarrow{false} L_6 \xrightarrow{next} L_7$

2. Multiple such iterations including infinite iterations, (+1)

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

4. $L_0 \xrightarrow{next} L_2 \xrightarrow{next} L_3 \xrightarrow{error} L_7$

   All other error executions. } (+1)

TOTAL : 4

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

1. p < 2 : (+1)

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

$L_4 \xrightarrow{call} L_1$, $L_1 \xrightarrow{return} L_5$

Multiple such iterations possible, (+1)

2. p >= 2 : (+1)

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

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

TOTAL : 7

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

Trace the actual execution of the program for the input p = 0.

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

$L_4 \xrightarrow[(+1)]{call} L_1$, $L_1 \xrightarrow[(+1)]{return} L_5$

(+1) true

$L_4 \xrightarrow[(+1)]{next} L_5$

(+1) call

$L_1 \xrightarrow[(+1)]{return} L_5$

(+1) next

L_3

(+1) false

$L_7 \xleftarrow[(+1)]{next} L_6$

TOTAL : 14