

---

## 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   expression assignment (+1)
      #end (+1)

```

(+1) indentation

(+1) locations

TOTAL: 6

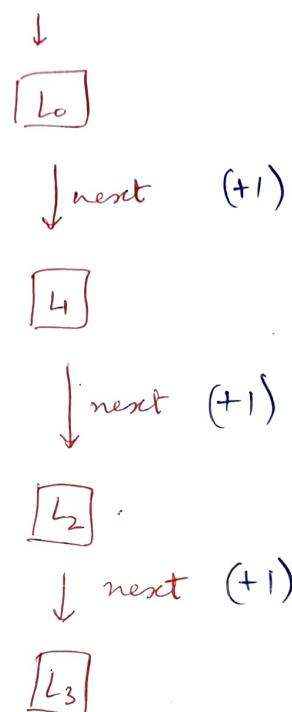
---

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

2.  $L_0 \xrightarrow{\text{next}} L_1 \xrightarrow{\text{error}} L_3$  }  
All other error executions. }  $(+1)$

TOTAL : 2

Trace the logically feasible executions of the program.

1.  $f$  is a number: (+1)

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

2.  $f$  is not a number: (+1)

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

TOTAL: 4

---

Trace the actual execution of the program for the input  $p = 5$ .

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

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.**

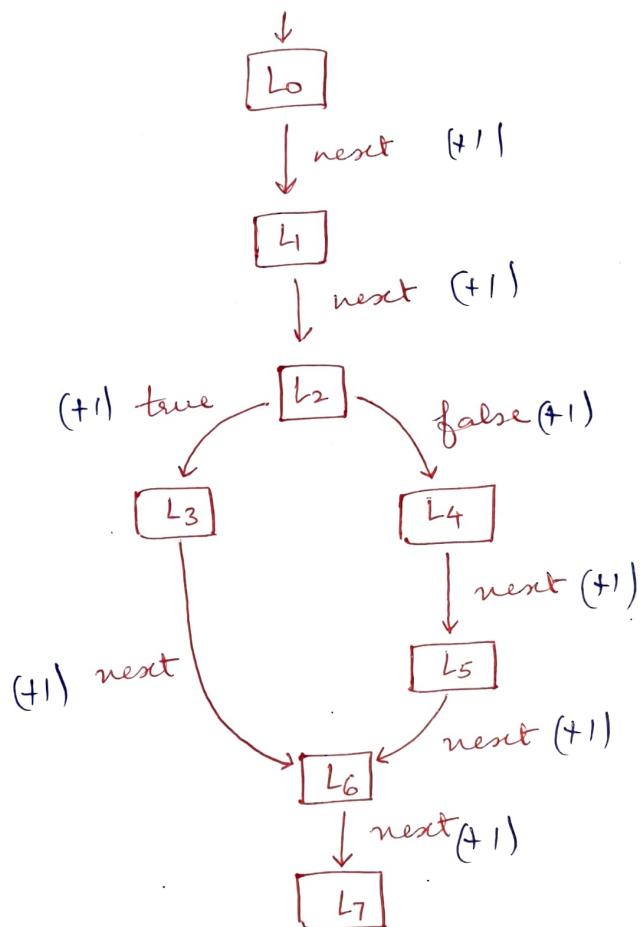
0 expression assignment (+1)  
 1 expression assignment (+1)  
 2 if:  
 3 expression assignment (+1)  
 4 else:  
 5 expression assignment (+1)  
 6 expression assignment (+1)  
 7 #end  
 TOTAL: 10  
 (+1) indentation  
 (+1) locations

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

i	next	true	false	call	return	error
0	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

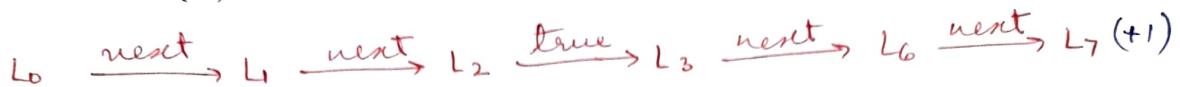
Trace the structurally feasible executions of the program.

1.  $L_0 \xrightarrow{\text{next}} L_1 \xrightarrow{\text{next}} L_2 \xrightarrow{\text{true}} L_3 \xrightarrow{\text{next}} L_6 \xrightarrow{\text{next}} L_7 (+1)$
2.  $L_0 \xrightarrow{\text{next}} L_1 \xrightarrow{\text{next}} L_2 \xrightarrow{\text{false}} L_4 \xrightarrow{\text{next}} L_5 \xrightarrow{\text{next}} L_6 \xrightarrow{\text{next}} L_7 (+1)$
3.  $L_0 \xrightarrow{\text{next}} L_1 \xrightarrow{\text{next}} L_2 \xrightarrow{\text{error}} L_7 \} (+1)$   
 All other error executions.

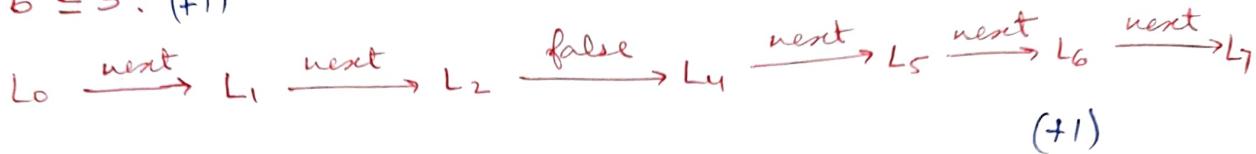
TOTAL: 3

Trace the logically feasible executions of the program.

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



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

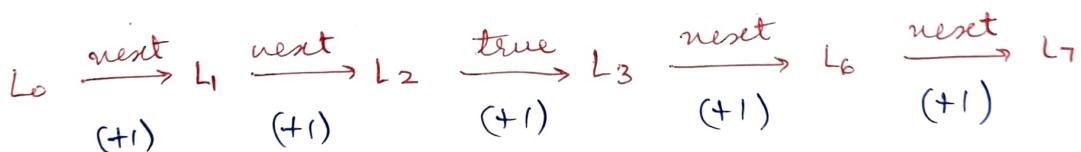


3.  $b$  is not a number : (+1)



TOTAL: 5

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



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 expression assignment (+1)  
 2 while:  
 3 expression assignment (+1)  
 4 expression assignment (+1)  
 5 continue (+1)  
 6 expression assignment (+1)  
 7 # end (+1)  
 (+1) indentation  
 (+1) locations

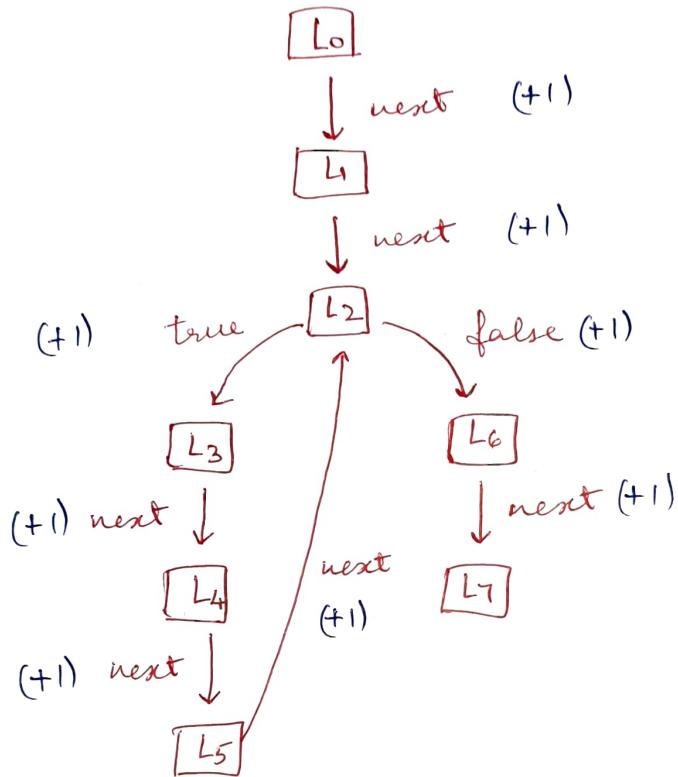
TOTAL: 10

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						

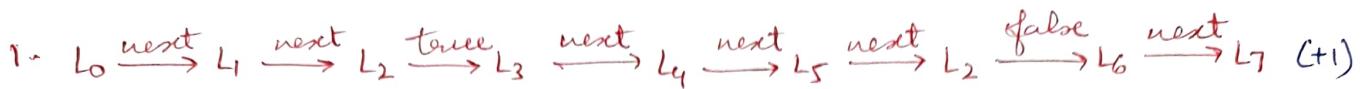
TOTAL: 14

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



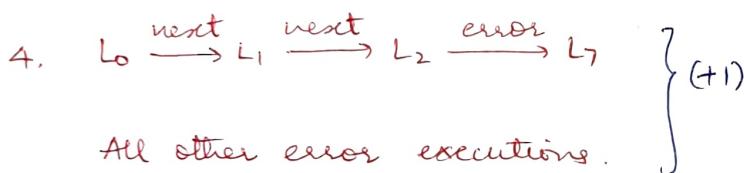
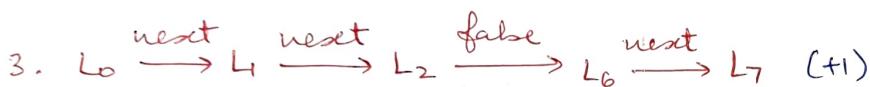
TOTAL : 8

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



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~~ would be enough to award 1 mark.



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.  $p = 1 \text{ or } c = 0$ : (+1)

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

2.  $p \neq 1 \text{ or } c \neq 0$ : (+1)

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

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

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

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

TOTAL: 7

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

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

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

TOTAL: 12

## Section 4

## Program

Write the structural abstraction of the program.

```

0 def div(a, b):
1     return a // b
2
3     x = read()
4
5     def func(p):
6         p = p - 2
7
8         k = div(5, p)
9
10    return k
11
12    m = div(4, x)
13
14    n = func(x)
15
16 # end

```

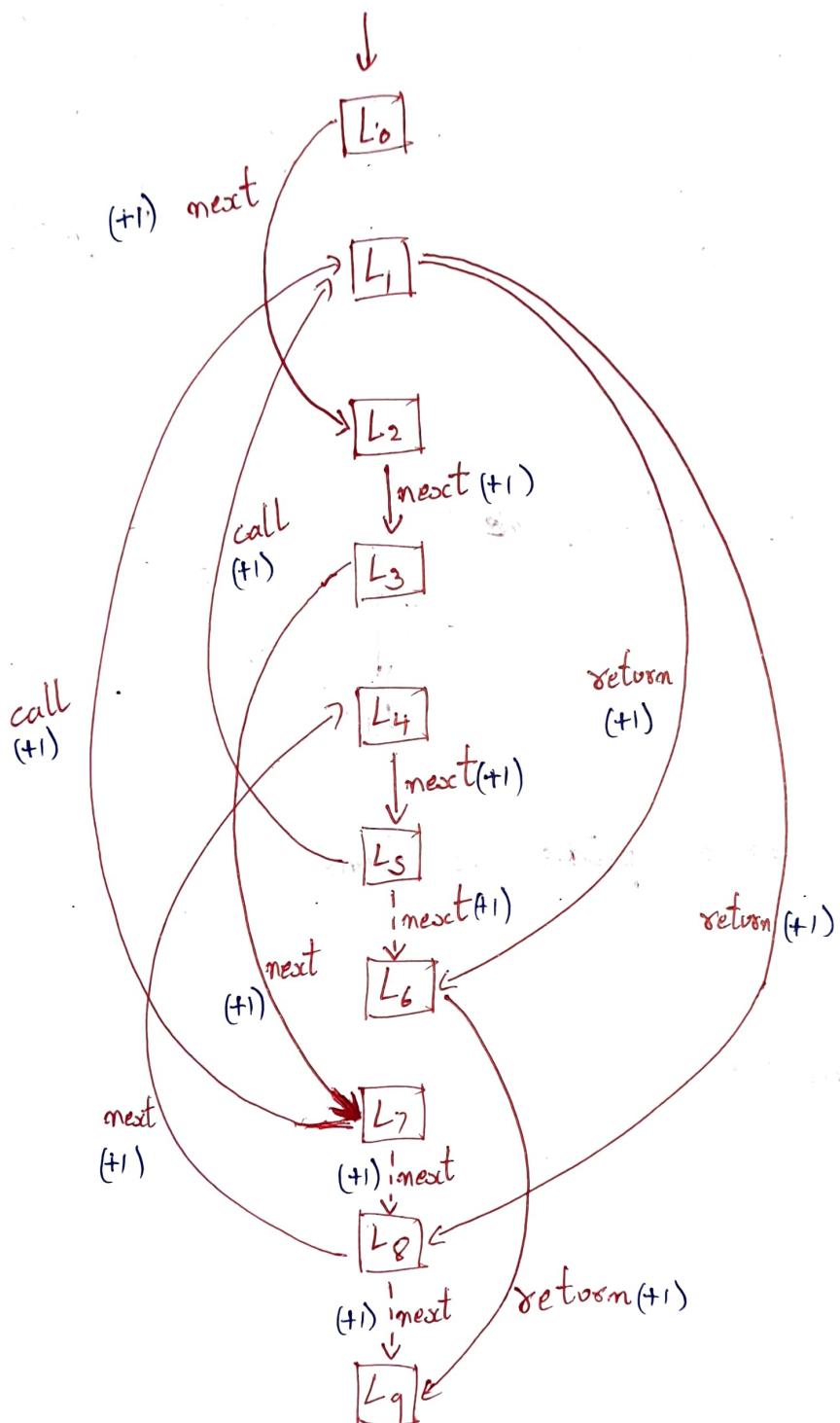
*(+1) locations**(+1) indentation*TOTAL: 12

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, 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						

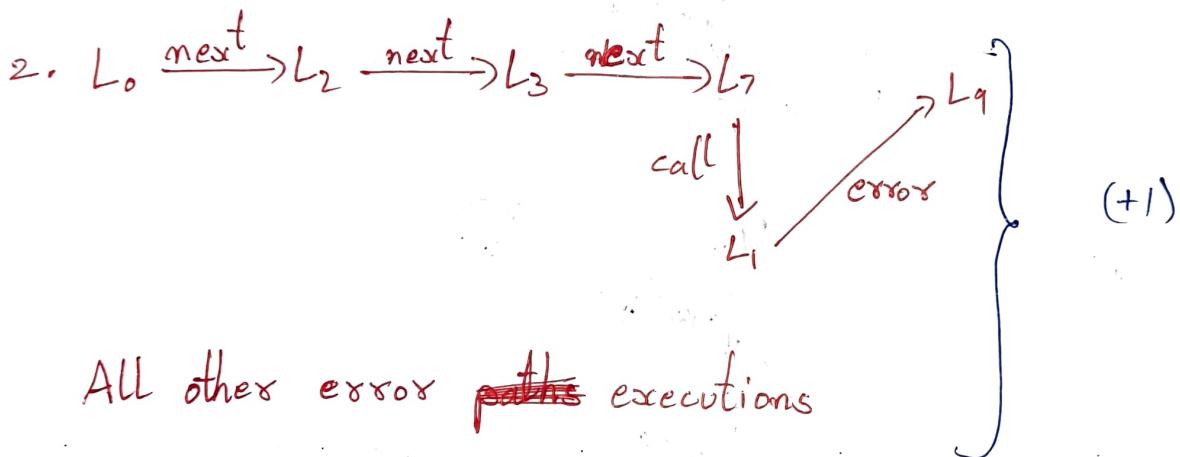
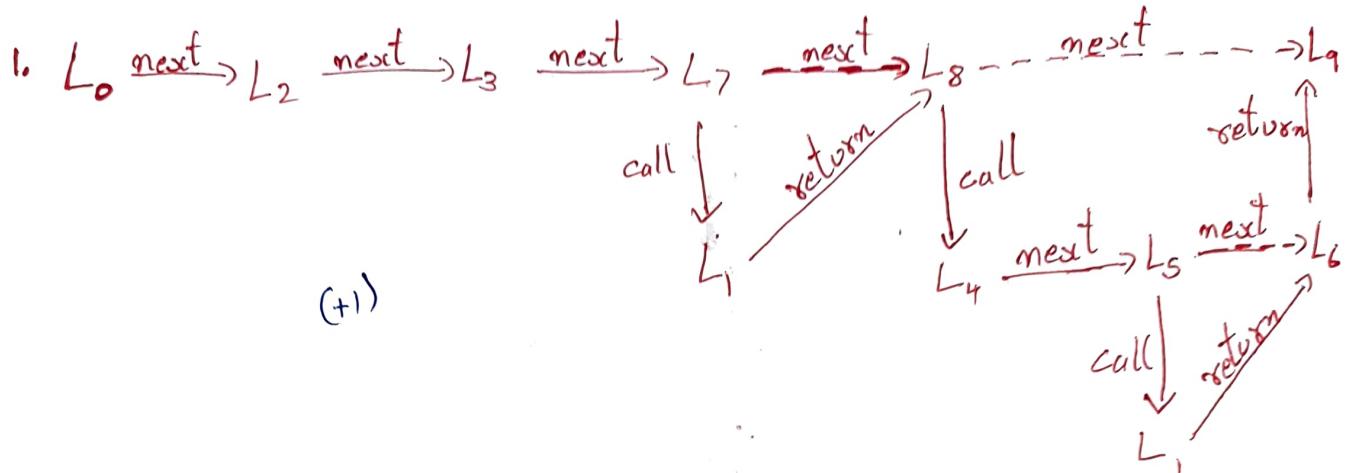
TOTAL: 19

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.

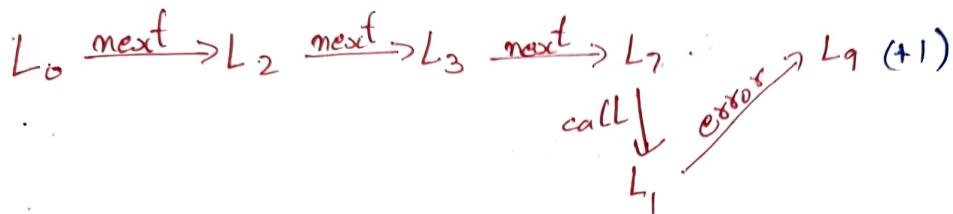


TOTAL: 2

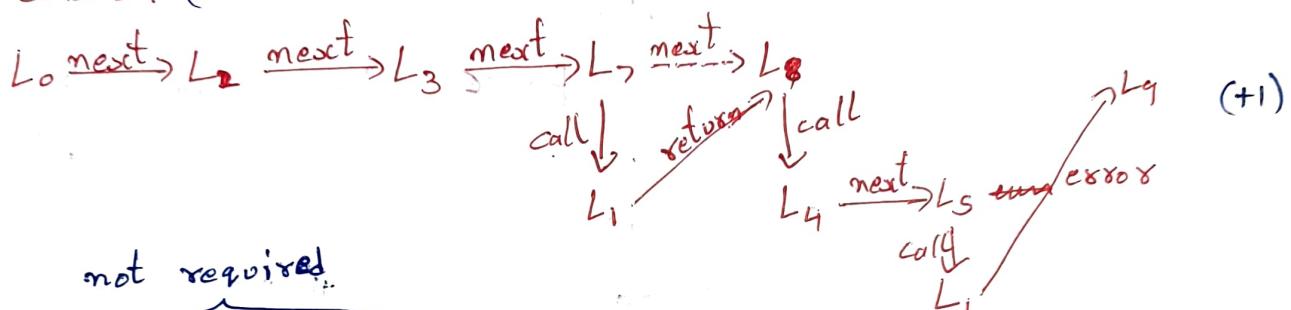
**Trace the logically feasible executions of the program.**

1.  $x = 0$  or  $x$  is not a number: (41)

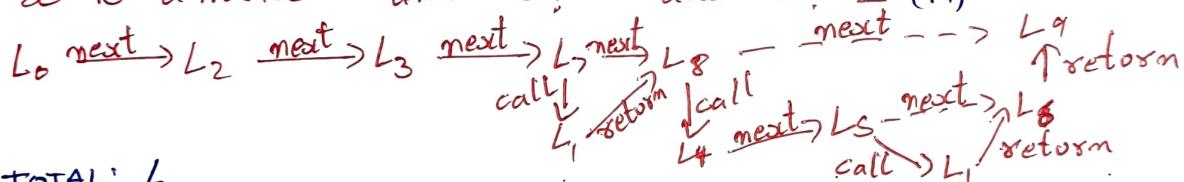
NOTE : +1 IF BOTH  
ARE MENTIONED



- $$2. \quad x = 2: (+1)$$

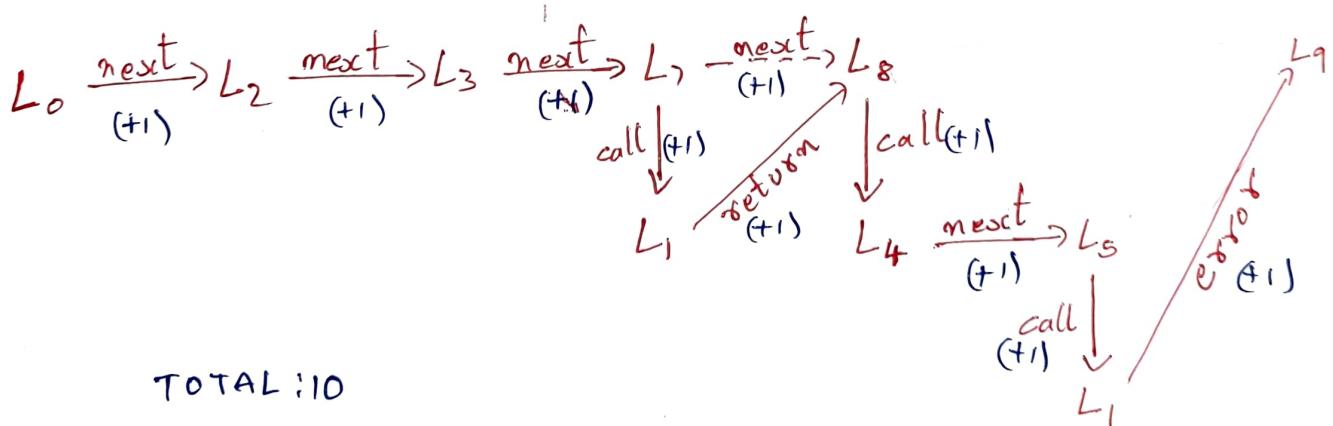


3.  $\infty$  is a number and  $\infty! = 0$  and  $\infty! = 2$ , (+i)



TOTAL: 6

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



## Section 5

## Program

```

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

```

Write the structural abstraction of the program.

0 def func:  
 1 return  
 2 expression assignment  
 3 while;  
 4 call func assignment  
 5 Continue  
 6 expression assignment  
 7 #end

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

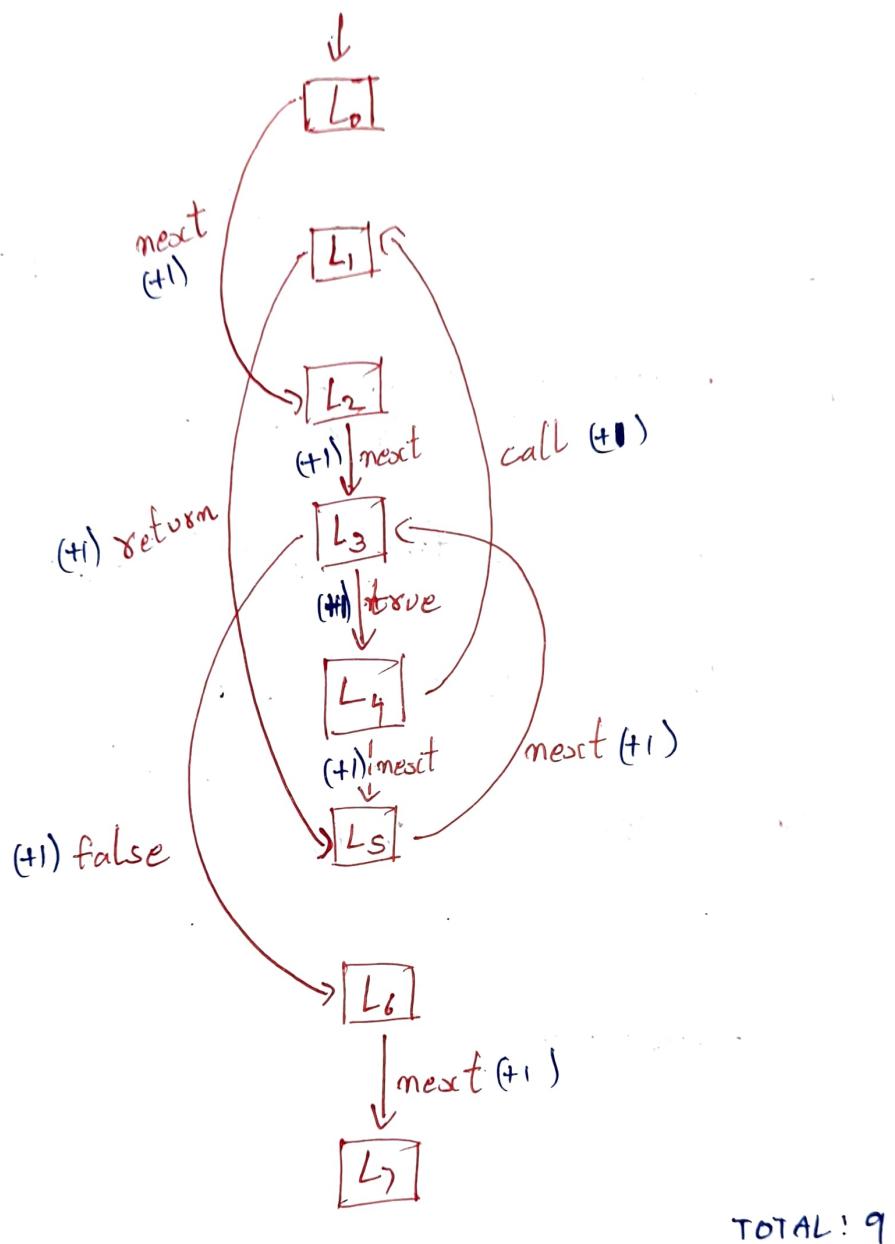
TOTAL: 10
(+1) locations  
(+1) indentation

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

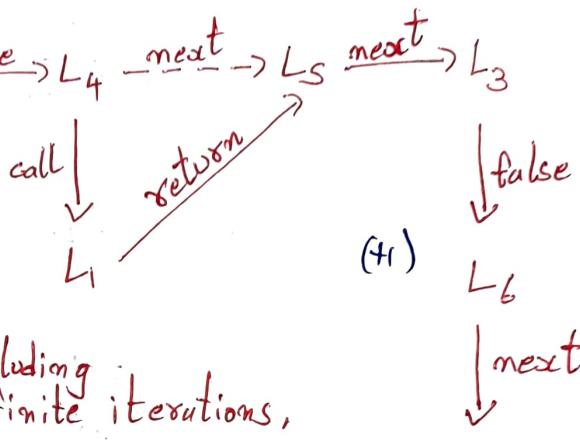
i	next	true	false	call	return	error
0	2					
1					{5}	7
2	3					7
3		4	6			7
4	5			1		7
5	3					
6	7					7
7						

TOTAL: 14

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



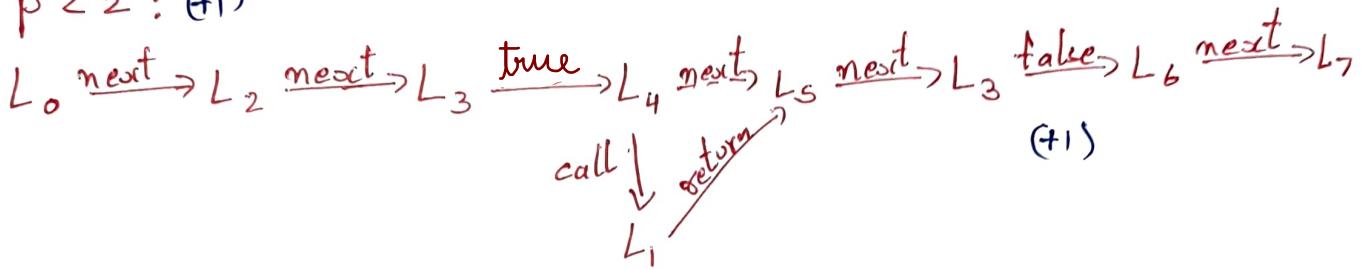
Trace the structurally feasible executions of the program. For executions which enter the while loop, showing one iteration is enough.

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

  2. Multiple such iterations including  
 (+1) infinite iterations,
  3.  $L_0 \xrightarrow{\text{next}} L_2 \xrightarrow{\text{next}} L_3 \xrightarrow{\text{false}} L_6 \xrightarrow{\text{next}} L_7$  (+1)
  4.  $L_0 \xrightarrow{\text{next}} L_2 \xrightarrow{\text{next}} L_3 \xrightarrow{\text{error}} L_7$  } (+1)
- All ~~A~~ other error execution,

TOTAL: 4

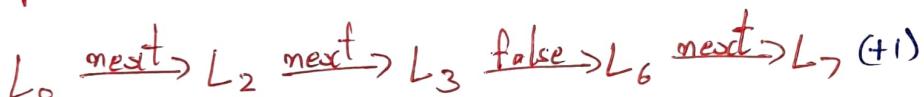
Trace the logically feasible executions of the program. For executions which enter the while loop, showing one iteration is enough.

1.  $p < 2$ : (+1)



Multiple such iterations possible. (+1)

2.  $p \geq 2$ : (+1)



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

TOTAL: 7



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

