Section 1

Program

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

Write the structural abstraction of the program.

0 expression assignment
1 expression assignment
2 expression assignment
3 #end

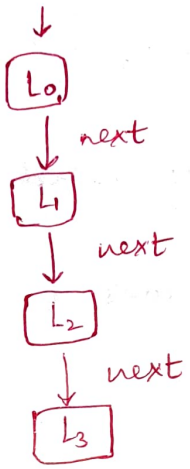
Total : 6

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

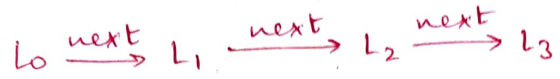
2. $L_0 \xrightarrow{error} L_3$ (or any valid error execution)

3. All other error executions.

Total : 2

Trace the logically feasible executions of the program.

1. x is a number:



2. x is not a number:

Total : 4

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

Total : 3

Section 2

Program

```
0
  x = read()
1
  if x > 10:
2
      y = 2
3
  else:
4
      y = x
5
      x = x * 2
6
  z = x - y
7
  # end
```

Write the structural abstraction of the program.

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

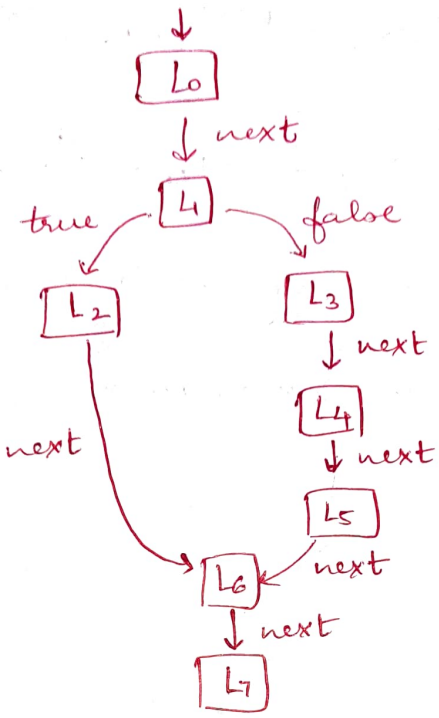
Total : 10

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	3			7
2	6					7
3	4					
4	5					7
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. $L_0 \xrightarrow{next} L_1 \xrightarrow{true} L_2 \xrightarrow{next} L_6 \xrightarrow{next} L_7$

2. $L_0 \xrightarrow{next} L_1 \xrightarrow{false} L_3 \xrightarrow{next} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_6 \xrightarrow{next} L_7$

3. $L_0 \xrightarrow{error} L_7$ (or any valid error execution)

4. All other error executions.

Total : 3

Trace the logically feasible executions of the program.

1. $x > 10$:

$L_0 \xrightarrow{next} L_1 \xrightarrow{true} L_2 \xrightarrow{next} L_6 \xrightarrow{next} L_7$

2. $x \leq 10$:

$L_0 \xrightarrow{next} L_1 \xrightarrow{false} L_3 \xrightarrow{next} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_6 \xrightarrow{next} L_7$

3. x is not a number:

$L_0 \xrightarrow{next} L_1 \xrightarrow{error} L_7$

Total : 6

Trace the actual execution of the program for the input x = 6.

$L_0 \xrightarrow{next} L_1 \xrightarrow{false} L_3 \xrightarrow{next} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_6 \xrightarrow{next} L_7$

Total : 6

Section 3

Program

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

Write the structural abstraction of the program.

0 expression assignment
1 expression assignment
2 while:
3 expression assignment
4 expression assignment
5 continue
6 expression assignment
7 #end

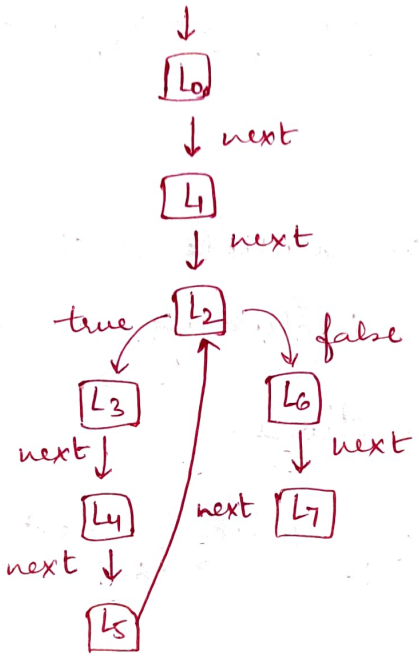
Total : 10

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 interation is enough.

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

2. Executions with multiple such iterations of the body block including an infinite number of iterations.

3. $L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{false} L_6 \xrightarrow{next} L_7$

4. $L_0 \xrightarrow{error} L_7$ (or any valid error execution).

5. All other error executions.

Total: 4

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

1. $i > 0$:

$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{true} L_3 \xrightarrow{next} L_4 \xrightarrow{next} L_5 \xrightarrow{next}$

$L_2 \xrightarrow{false} L_6 \xrightarrow{next} L_7$

Multiple such possible executions based on the value of i

2. $i \leq 0$:

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

3. i is not a number:

$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{error} L_7$

Total : 7

Trace the actual execution of the program for the input `i = 2`.

$L_0 \xrightarrow{next} L_1 \xrightarrow{next} L_2 \xrightarrow{true} L_3 \xrightarrow{next} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_2 \xrightarrow{true}$

$L_3 \xrightarrow{next} L_4 \xrightarrow{next} L_5 \xrightarrow{next} L_2 \xrightarrow{false} L_6 \xrightarrow{next} L_7$

Total : 12

Section 4

Program

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

Write the structural abstraction of the program.

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

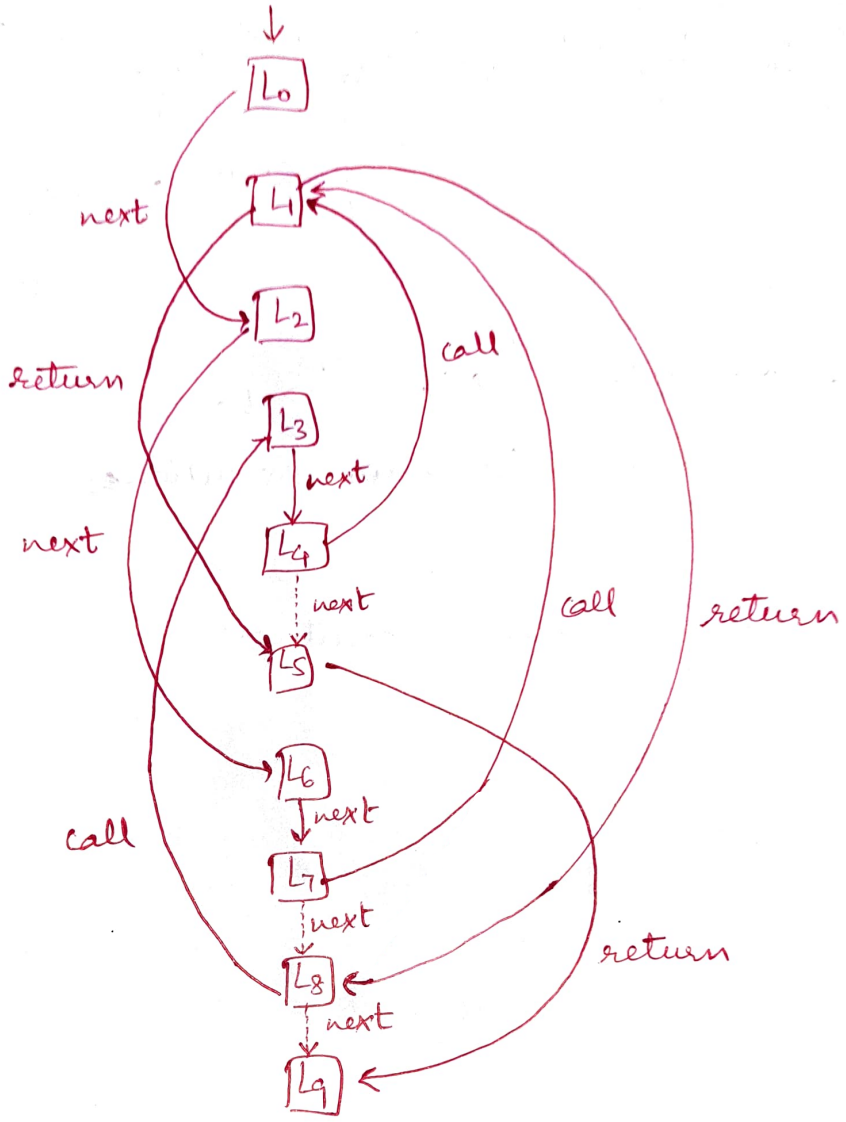
Total : 12

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

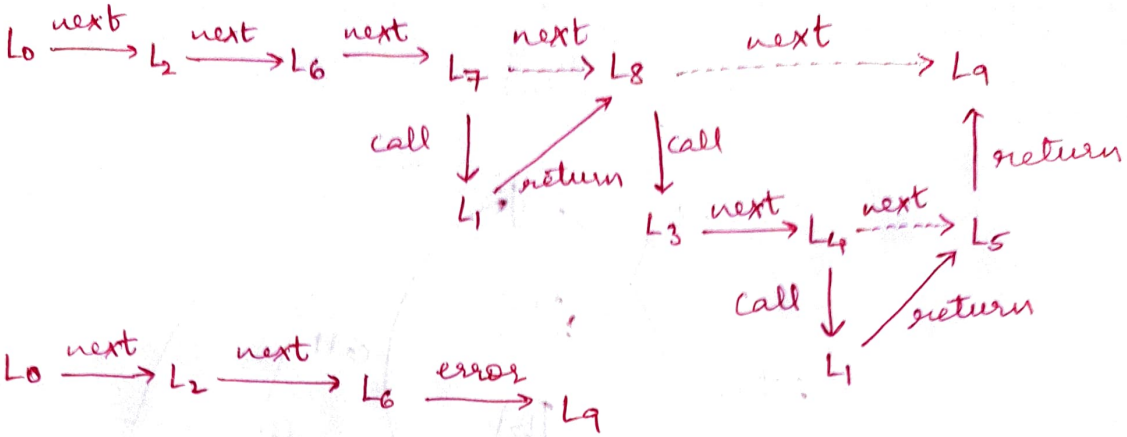
Total : 19

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. 

2. L_0 next L_2 next L_6 error L_9

   (or any valid error execution)

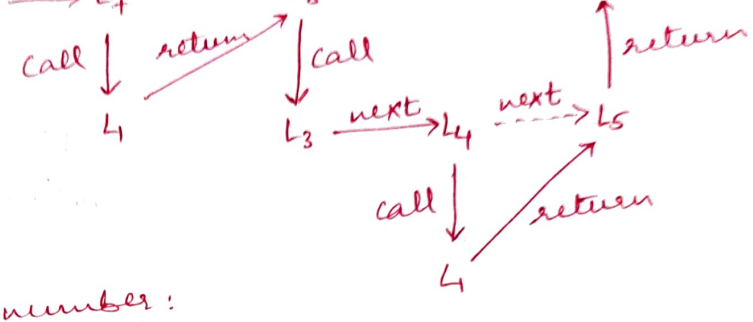
3. All other error executions

Total : 2

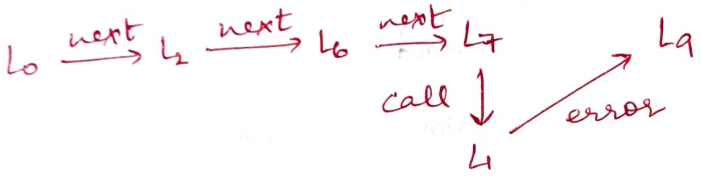
Trace the logically feasible executions of the program.

m is a number :

1. $L_0 \xrightarrow{next} L_2 \xrightarrow{next} L_6 \xrightarrow{next} L_7 \xrightarrow{next} L_8 \xrightarrow{next} L_9$

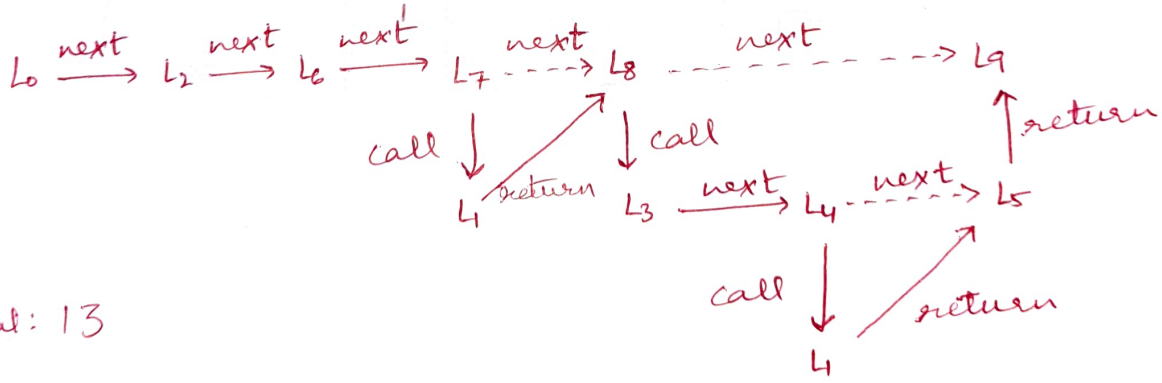


2. m is not a number :



Total : 4

Trace the actual execution of the program for the input `m = 3`.



Total: 13

Section 5

Program

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

Write the structural abstraction of the program.

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

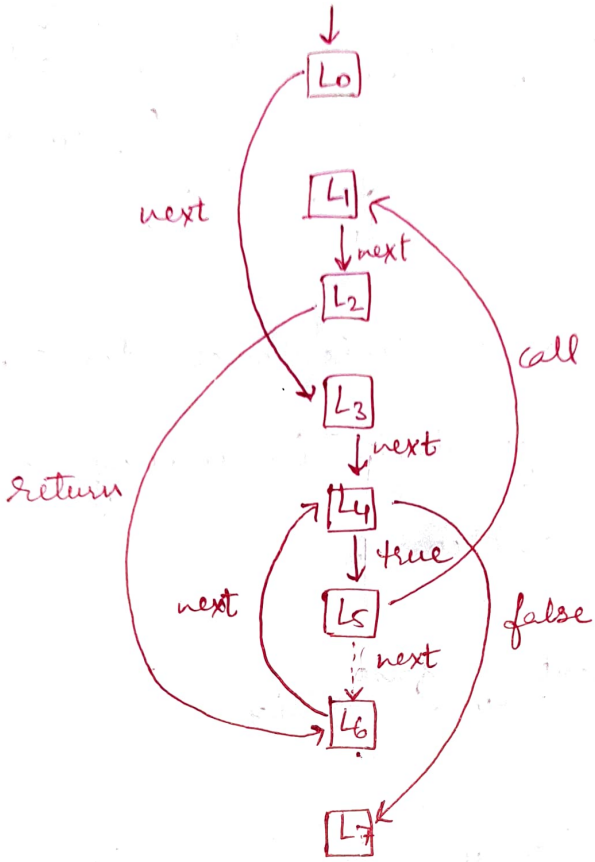
Total : 10

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

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

1. $L_0 \xrightarrow{next} L_3 \xrightarrow{next} L_4 \xrightarrow{true} L_5 \dashrightarrow L_6 \xrightarrow{next} L_4 \xrightarrow{false} L_7$

   $L_4 \xrightarrow{call} L_1 \xrightarrow{next} L_2 \xrightarrow{return} L_4$

2. Executions with multiple such iterations of the while body block including an infinite number of iterations.

3. $L_0 \xrightarrow{next} L_3 \xrightarrow{next} L_4 \xrightarrow{false} L_7$

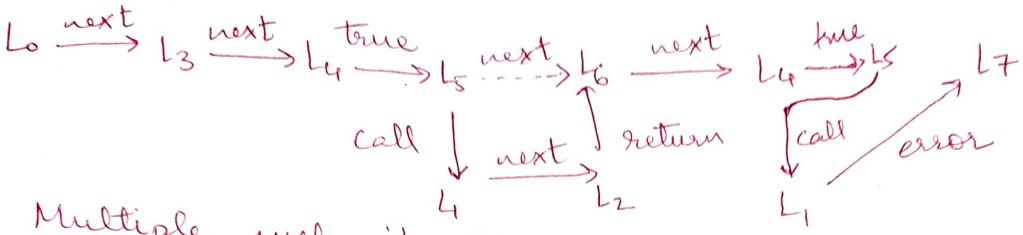
4. $L_0 \xrightarrow{next} L_3 \xrightarrow{error} L_4$ (or any valid error execution)

5. All other error executions.

Total : 4

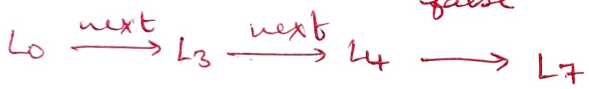
Trace the logically feasible executions of the program. For executions which enter the while loop, showing one interation is enough.

1. $i \geq 0$:

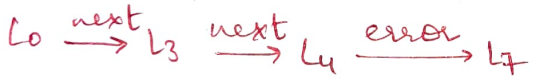


Multiple such iterations of the while body block based on the value of i.

2. $i < 0$:

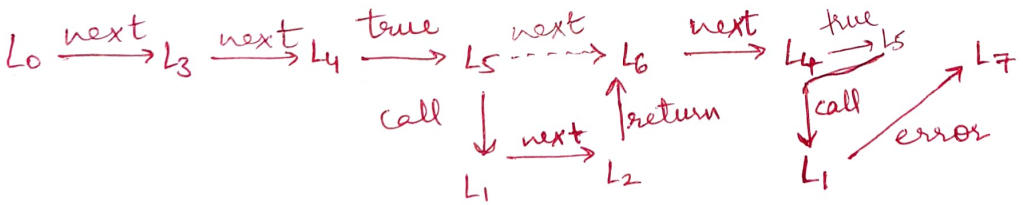


3. i is not a number:



Total : 6

Trace the actual execution of the program for the input `i = 1`.



Total : 11