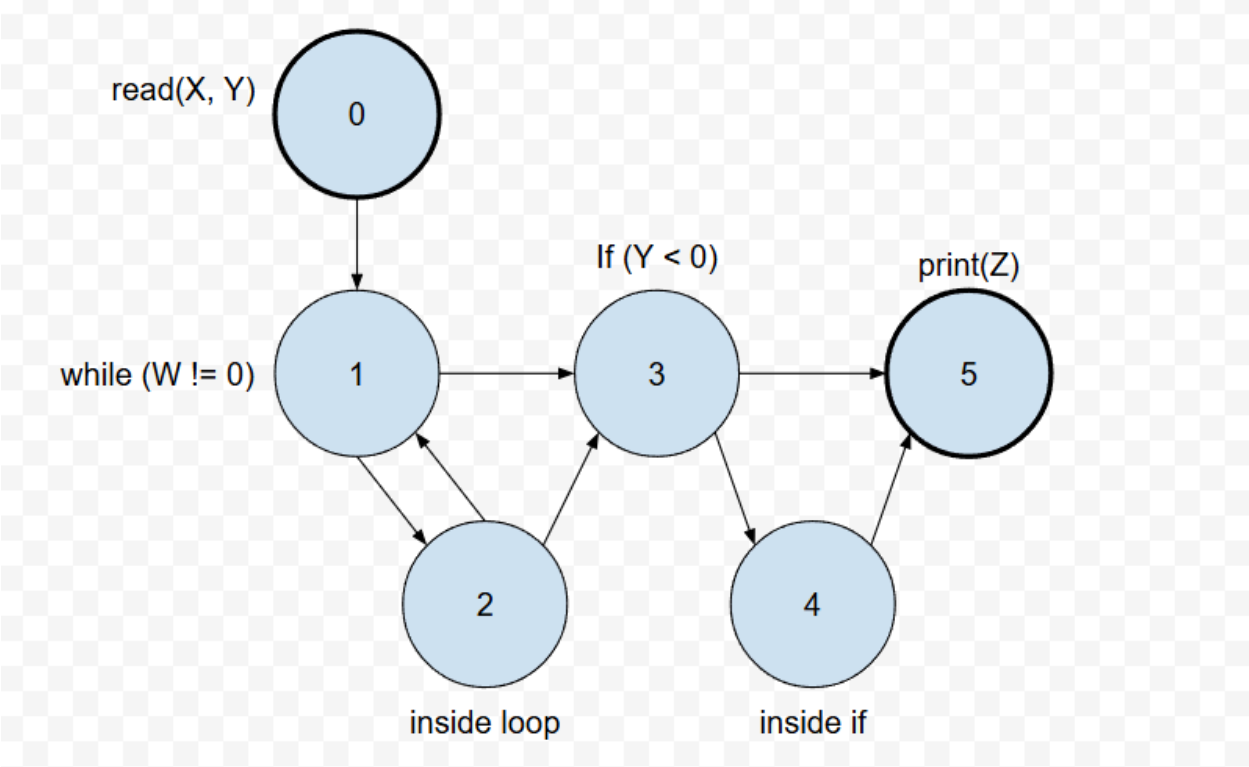
Q1:

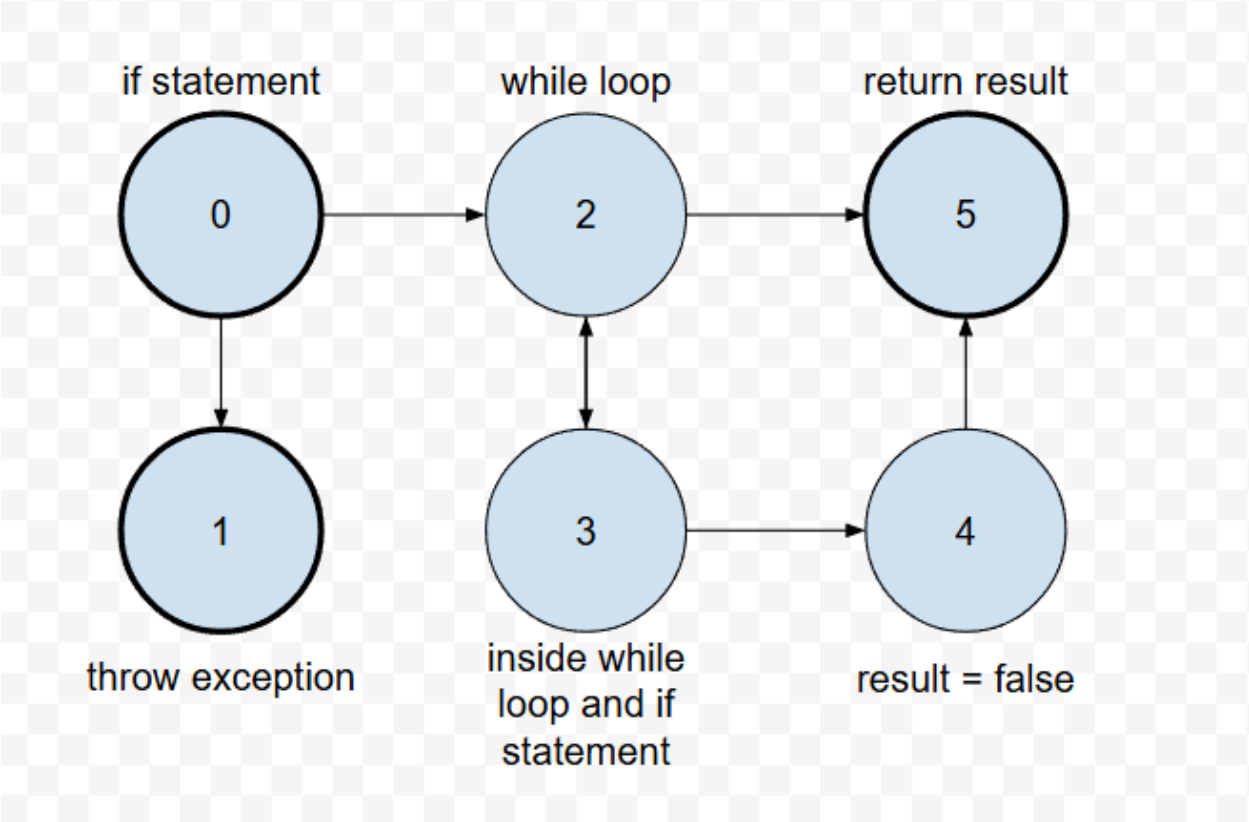


1. There are no infeasible paths. All parts of the code can be reached and no conditions lead to unreachable blocks of code.

2. Node Coverage:
   - TR = {0, 1, 2, 3, 4, 5}
   - Test Paths = {[0, 1, 2, 1, 3, 5], [0, 1, 3, 4, 5]}

3. Edge Coverage:
   - TR = {(0, 1), (1, 2), (2, 1), (1, 3), (2, 3), (3, 4), (3, 5), (4, 5)}
   - Test Paths = {[0, 1, 2, 1, 2, 3, 5], [0, 1, 3, 4, 5]}

Q2:
1. CFG



2.
TR(NC) = {0, 1, 2, 3, 4, 5}
TR (EC) = {(0, 1), (0, 2), (2, 3), (3, 4), (4, 5), (2, 5)}
TR (EPC) = {(0, 2, 5), (0, 2, 3), (2, 3, 4), (2, 3, 2), (3, 2, 3), (3, 4, 5), (3, 2, 5)}

3.
Node Coverage but not Edge Coverage: A single test case that only goes through a linear path without taking all branches.

- {null, “sa”}: null goes from 0 to 1. “sa” where right > left so it doesn’t execute the while loop. It goes from 0, 2, and 5.

Edge Coverage but not Edge Pair Coverage.

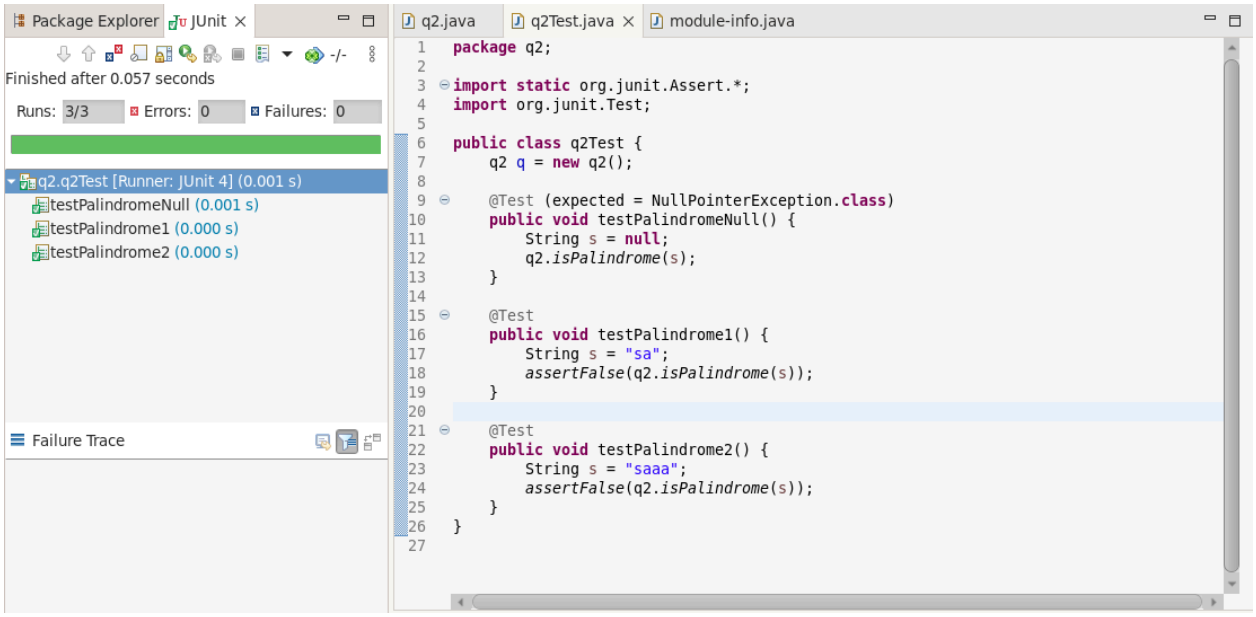
- not possible

Edge Pair Coverage

- {“saaa”}

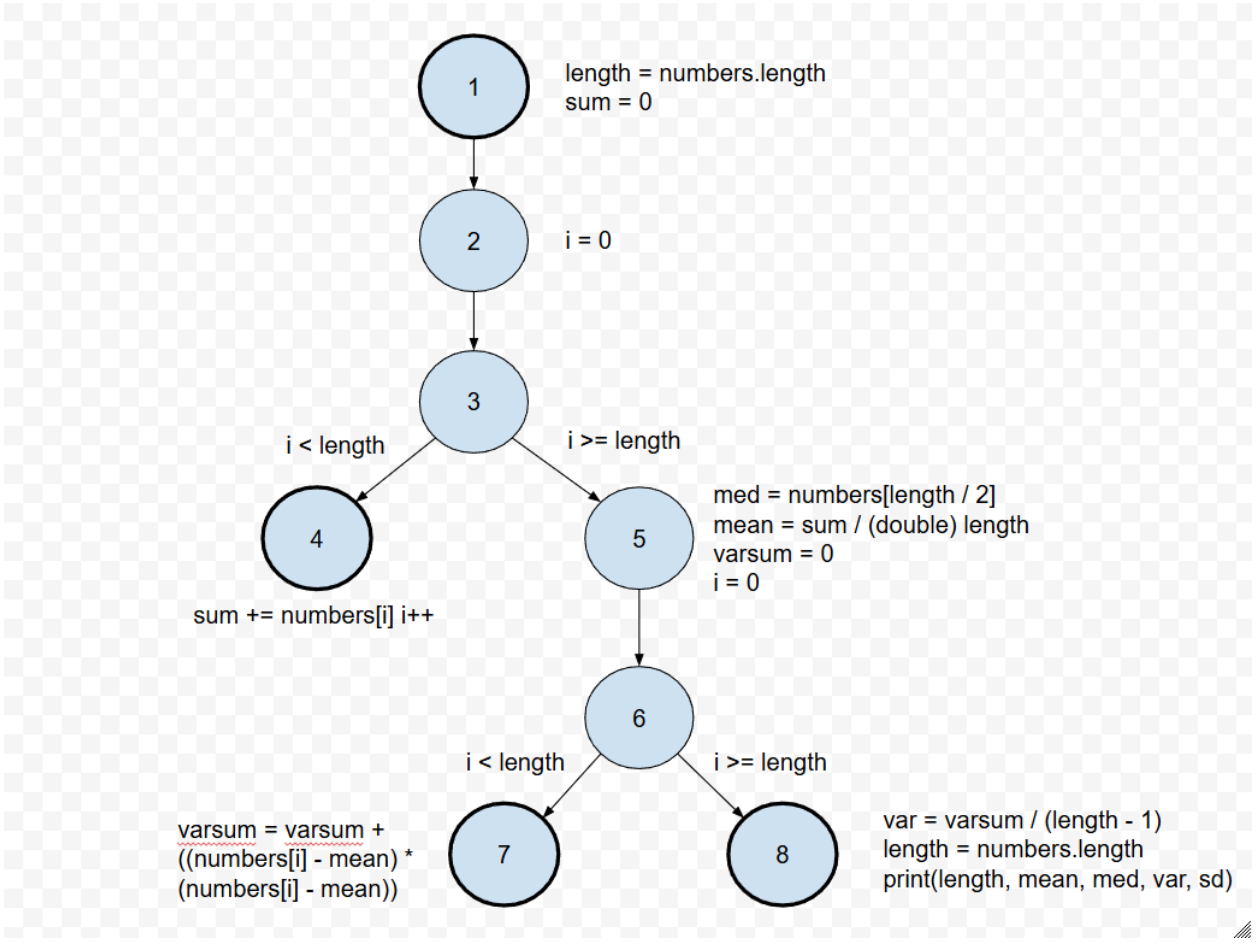
4.
Prime Path Coverage are paths that do not repeat nodes except possibly at the start or the end.
TR(PPC): 0→1, 0→2→3→4→5, 0→2→5
PPC: {null, “sa”, “saaa”}

5. Screenshot of the JUnit test class

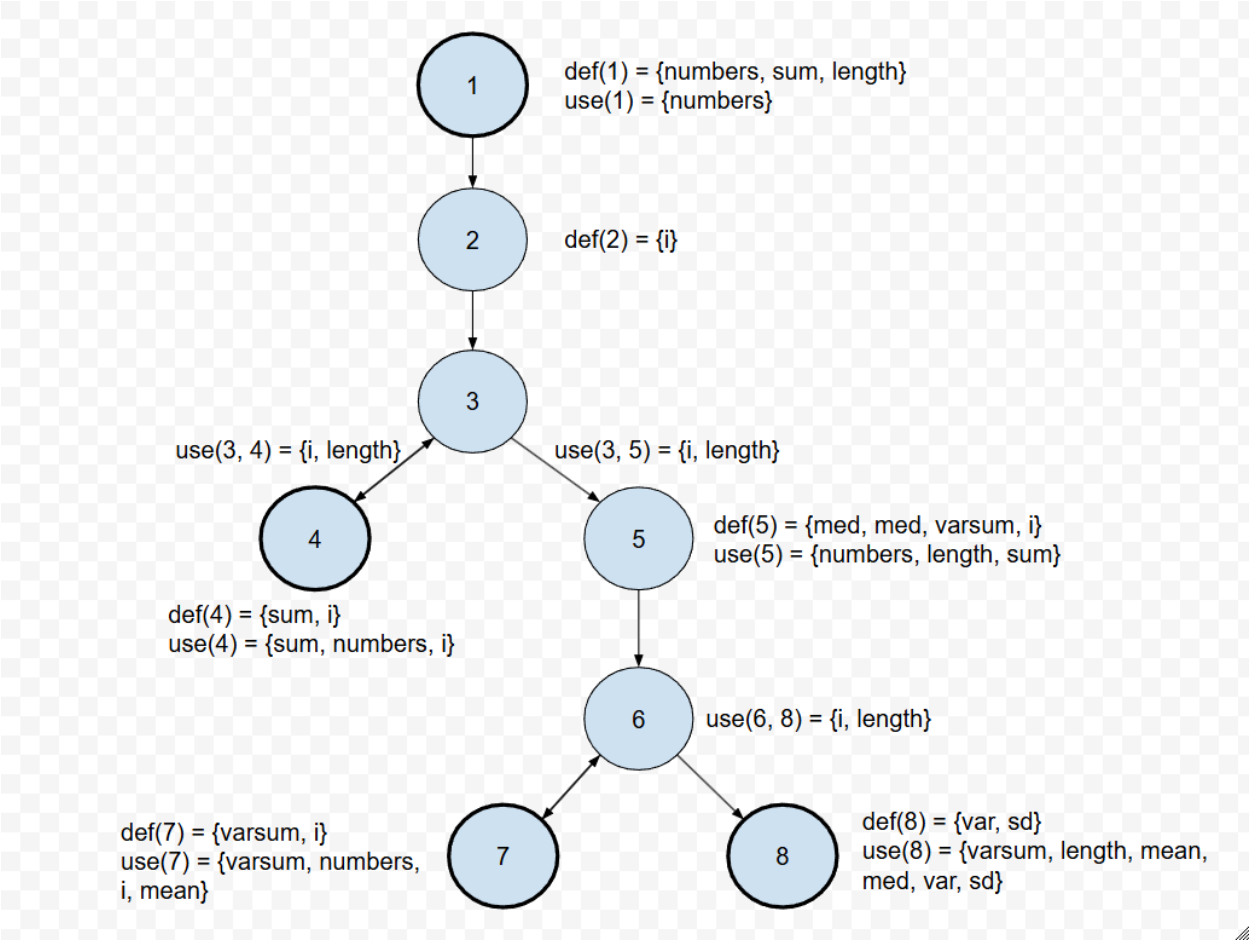


Q3:

1. CFG



DFG



2. du pairs

numbers	14, 15, 17
sum	14, 15, 45
length	13, 15, 16, 18
i	23, 24, 25, 26, 56, 57
var	
varsum	78
mean	58, 57
med	58
sd	

sd and var has no node pair because sd was only used and initialized in node 8 and var was only used and initialized in node 7. Since they have no du pairs or node pairs, this means that they also don't have du paths.

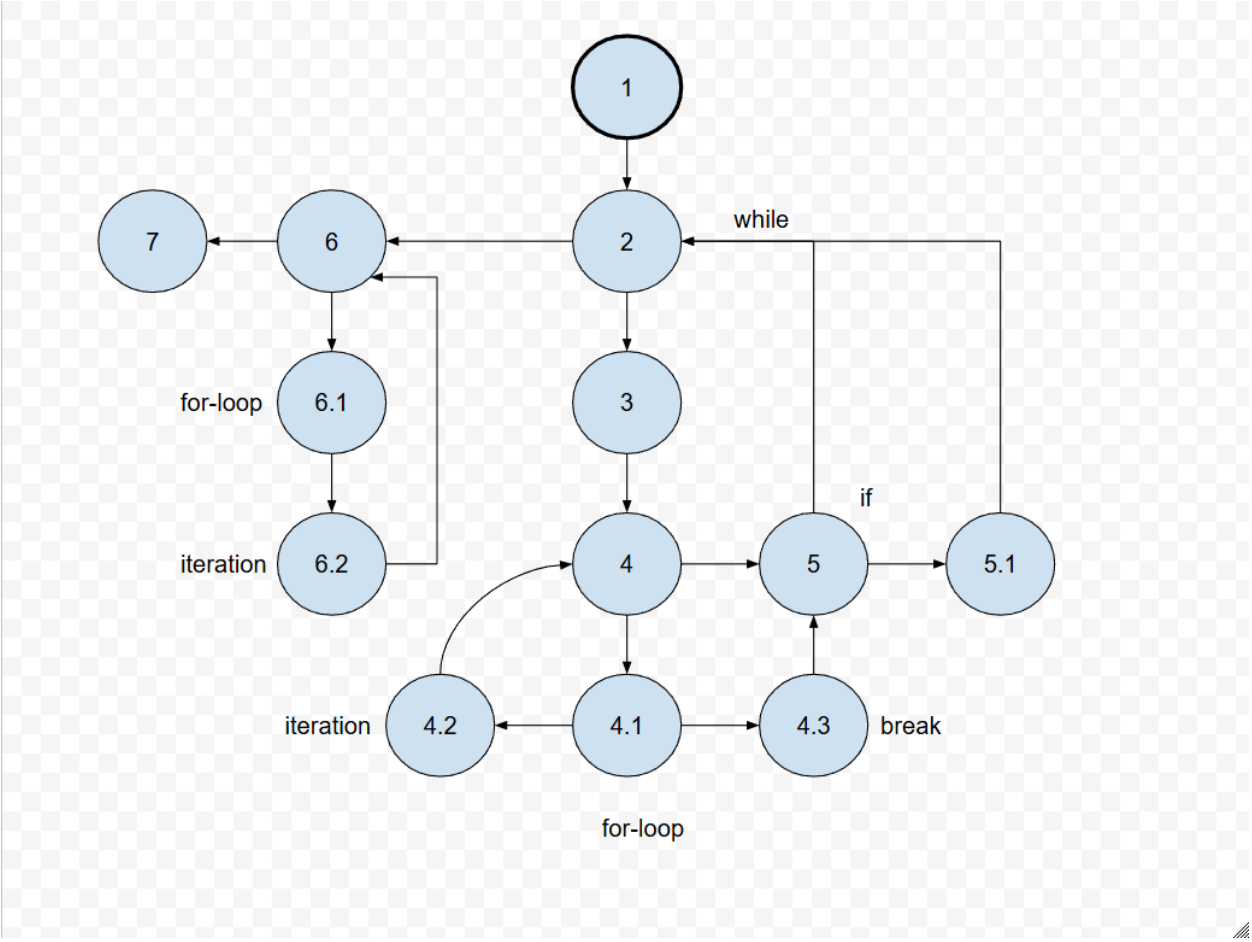
3. du paths

numbers	[1, 2, 3, 4] [1, 2, 3, 5] [1, 2, 3, 5, 6, 7]
sum	[1, 2, 3, 4] [1, 2, 3, 5] [4, 3, 5]
length	[1, 2, 3, 5] [1, 2, 3, 5, 6, 8] [1, 2, 3, 4] [1, 2, 3, 5, 6, 7] [1, 2, 3, 5, 6, 8]
i	[2, 3, 4] [2, 3, 5] [4, 3, 4] [4, 3, 5] [5, 6, 7] [5, 6, 8] [7, 6, 7] [7, 6, 8]
var	
varsum	[7, 6, 8]
mean	[5, 6, 7] [5, 6, 8]
med	[5, 6, 8]
sd	

4. Test cases: numbers (20), numbers (5, 10, 15)

5. If a value of 0 is provided to the program, the program will return an error line as the line within the code evaluates some number divided by length. Since the length is 0 and dividing by 0, this would return an error.

Q4:
1. CFG



2. if n=0, numPrimes < n returns false

3. T:{1, 2, 3, 4, 4.1, 4.2, 4, 5, 5.1, 2, 3, 4, 4.1, 4.3, 5, 2, 6, 6.1, 6.2, 6, 6.1, 6.2, 6, 6.1, 6.2, 6, 6}
input is 3

4. Screenshot of the JUnit test cases