Use the method printPrimes() for questions a–f below. A compilable version is attached: PrintPrimes.java. A line-numbered version is attached: PrintPrimes.num.

1. Draw the control flow graph for theprintPrimes() method.
2. Consider test cases t1= (n= 3) and t2= (n= 5). Although these tour the same primepaths in printPrimes(), they do not necessarily find the same faults. Design a simplefault that t2 would be more likely to discover than t1 would.
3. For printPrimes(), find a test case such that the corresponding test path visits the edge that connects the beginning of the *while* statement to the *for* statement that appears after the *while* loop, **without** going through the body of the *while* loop.
4. List the test requirements for Node Coverage, Edge Coverage, and Prime Path Coverage.
5. List test paths that achieve Node Coverage but not Edge Coverage on the graph.
6. List test paths that achieve Edge Coverage but not Prime Path Coverage on the graph.

a-

Considering we are using the fixed version of computePrimes method, for n <= 0 we get one characteristic and for n > 0 we get another characteristic.

b- Instead of using

if (isPrime && (number % 10 != 9)) // FAULT

we can use

if (isPrime && (number % 5 != 0)) // FAULT

to create a fault that would be discovered the second input and not the first input.

c- n <= 0 would execute the while statement but not go inside it since the condition is false.

d- For node and edge coverage we need to test at least one value from all blocks. So we need to test at least one positive and one negative numbers.For prime path coverage we need to test at least one positive number.

e- It is no possible to achieve node coverage without edge coverage since we need to use the edge to get to the nodes.

f- It is not possible to achieve edge coverage without prime path coverage since we need the edges of prime path for edge coverage.

a-