while

numPrimes > n

numPrimes < n

if 2

for 3

isDivisible

!showAll

showAll

i > numPrimes - 1

i <= numPrimes - 1

if 7

4

for 5

if

isDivisible

listOfPrimes = new int[1];
listOfPrimes[0] =
primes[numPrimes - 1]

thisPrime == 0

thisPrime != 0

if 6

!isPrime

return listOfPrimes 10

9

isPrime

primes[numPrimes]
= curPrime;

8

numPrimes++;

#2.) Test Case from first while to next if to end:

N1 -> N2 -> N4 -> N10

(numPrimes > n)   ->   (showAll = false) -> listofPrimes[0] = primes[numPrimes – 1]
-> (return listOfPrimes)
Input Values: n = 0, showAll = false.

3.) Test Paths that achieve edge coverage w/o prime path:

*Test Requirements: (1,2),(1,3), (2,4),(2,5),(3,6),(3,7),(4,10), (5,9), (5,10), (6,1),(6,8),
(7,3),(7,6), (8,1),(9,5)

a.) Path #1: (N1,N3,N7,N3,N6,N8,N1,N3,N7,N6,N1,N2,N4,N10)

*Edges Covered: (1,2),(1,3),(2,4),(3,6),(3,7),(4,10),(6,1),(6,8),(7,3),(7,6),(8,1)
*Input values: n = 3, showAll = false

b.) Path #2:
(N1,N3, N7,N3,N6,N8,N1,N3,N7,N6,N1,N2,N5,N9,N5,N10)
*Edges Covered: (1,2),(1,3),(2,5),(3,6),(3,7),(4,10),(5,9),(5,10),(6,1),(6,8),(7,3),(7,6),(8,1)(9,5)

*Input values: n = 3, showAll = true