2. **[20 marks]** Consider the following function that prints the first n primes.

```
private static void printPrimes (int n)
{
  int curPrime; // Value currently considered for primeness
  int numPrimes; // Number of primes found so far.
  boolean isPrime; // Is curPrime prime?
  int [] primes = new int [MAXPRIMES]; // The list of prime numbers.

  // Initialize 2 into the list of primes.
  primes [0] = 2;
  numPrimes = 1;
  curPrime = 2;
  while (numPrimes < n)
  {
    curPrime++; // next number to consider ...
    isPrime = true;
    for (int i = 0; i <= numPrimes-1; i++)
    { // for each previous prime.
      if (isDivisible (primes[i], curPrime))
      { // Found a divisor, curPrime is not prime.
        isPrime = false;
        break; // out of loop through primes.
      }
    }
    if (isPrime)
    { // save it!
    primes[numPrimes] = curPrime;
    numPrimes++;
    }
  } // End while

  // Print all the primes out.
  for (int i = 0; i <= numPrimes-1; i++)
  {
    System.out.println ("Prime: " + primes[i]);
  }
}
```

Draw the Control Flow Graph for this function. Clearly indicate what are the segments (you can do this on the code if you prefer).

The 12 segments for printPrimes (each worth 1 mark) are shown below. The Control Flow Graph is shown next (8 marks). Each incorrect edge is minus two marks. Segment Z denotes the end of the execution of printPrimes.

Segment A: Lines 1-11

Segment B: Line 12

Segment C: Lines 14-15 plus the initialization of the for loop.

Segment D: `i <= numPrimes - 1`

Segment E: Lines 17-18

Segment F: Lines 20-21

Segment G: `i++`

Segment H: Line 24

Segment I: Lines 26-27

Segment J: The initialization od the for loop in line 32

Segment K: `i <= numPrimes - 1`

Segment L: Line 34 plus `i++`

$$\begin{tikzcd}
 & & A \arrow[d] & & & \\
 & J \arrow[d] & B \arrow[l] \arrow[r] & C \arrow[r] & D \arrow[dl] \arrow[d] & G \arrow[l] \\
Z & K \arrow[l] \arrow[d, shift right] & & H \arrow[ul] \arrow[d] & E \arrow[ur] \arrow[d] & \\
 & L \arrow[u, shift right] & & I \arrow[uul] & F \arrow[ul] &
\end{tikzcd}$$

3. **[30 marks]** For the function of the previous question, indicate and explain briefly how you derived

   (a) A minimal test suite that achieves 100% segment coverage [10 marks].

   Segments A and B will be covered with any test case. Similarly, any test case where n is larger than 0 will cover segments J, K, and L. A value for n larger than 1 will ensure that we enter the loop, which will cover segments C,D,E,G,H,I (notice that we cannot exit the loop unless we visit segment I). So, we simply need to select a test case that will take us to segment F. This means that as we are searching for prime numbers, we encounter a number that is not prime. For this to happen, we need n to be 3. This will discover the prime numbers 2, 3, and 5, and will pass over 4.

   Therefore, our minimal test suite consists of one test case, n=3.

   (b) A minimal test suite that achieves 100% branch coverage [10 marks].

   The test suite that achieves segment coverage, also achieves branch coverage, so there is no need for additional test cases.