Homework #1

1. (50 points) Answer the following questions about the graph I.

   (a) Draw the graph using DOT language

   A. You can use `null0 [shape=point]` to draw an arrow to an initial node

   B. You can visualize your graph using http://www.webgraphviz.com/ or graphviz utilities

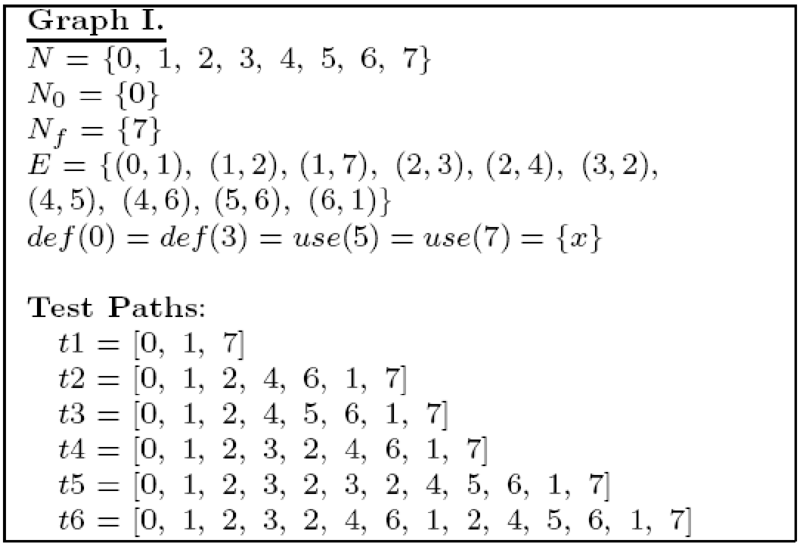
   (b) List all of the du-paths with respect to x. (Note: Include all du-paths, even those that are subpaths of some other du-path).

   (c) For each test path, determine which du-paths that test path **du-**tours (i.e., satisfying the def-clear requirement). For this part of the exercise, you should consider both direct touring and sidetrips. Hint: A table is a convenient format for describing this relationship.

   (d) List a minimal test set that satisfies all defs coverage with respect to x. (Direct tours only.) Use the given test paths.

   (e) List a minimal test set that satisfies all uses coverage with respect to x. (Direct tours only.) Use the given test paths.

   (f) List a minimal test set that satisfies all du-paths coverage with respect to x. (Direct tours only.) Use the given test paths.



Graph I.

$N = \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7\}$

$N_0 = \{0\}$

$N_f = \{7\}$

$E = \{(0,1),\ (1,2),\ (1,7),\ (2,3),\ (2,4),\ (3,2),$
$(4,5),\ (4,6),\ (5,6),\ (6,1)\}$

$def(0) = def(3) = use(5) = use(7) = \{x\}$

Test Paths:

$t1 = [0,\ 1,\ 7]$

$t2 = [0,\ 1,\ 2,\ 4,\ 6,\ 1,\ 7]$

$t3 = [0,\ 1,\ 2,\ 4,\ 5,\ 6,\ 1,\ 7]$

$t4 = [0,\ 1,\ 2,\ 3,\ 2,\ 4,\ 6,\ 1,\ 7]$

$t5 = [0,\ 1,\ 2,\ 3,\ 2,\ 3,\ 2,\ 4,\ 5,\ 6,\ 1,\ 7]$

$t6 = [0,\ 1,\ 2,\ 3,\ 2,\ 4,\ 6,\ 1,\ 2,\ 4,\ 5,\ 6,\ 1,\ 7]$

2. (50 points) Use the following method printPrimes() which prints `n` small prime numbers with a given input `n` for questions a-e below. Answer the question based on the given control flow graph.

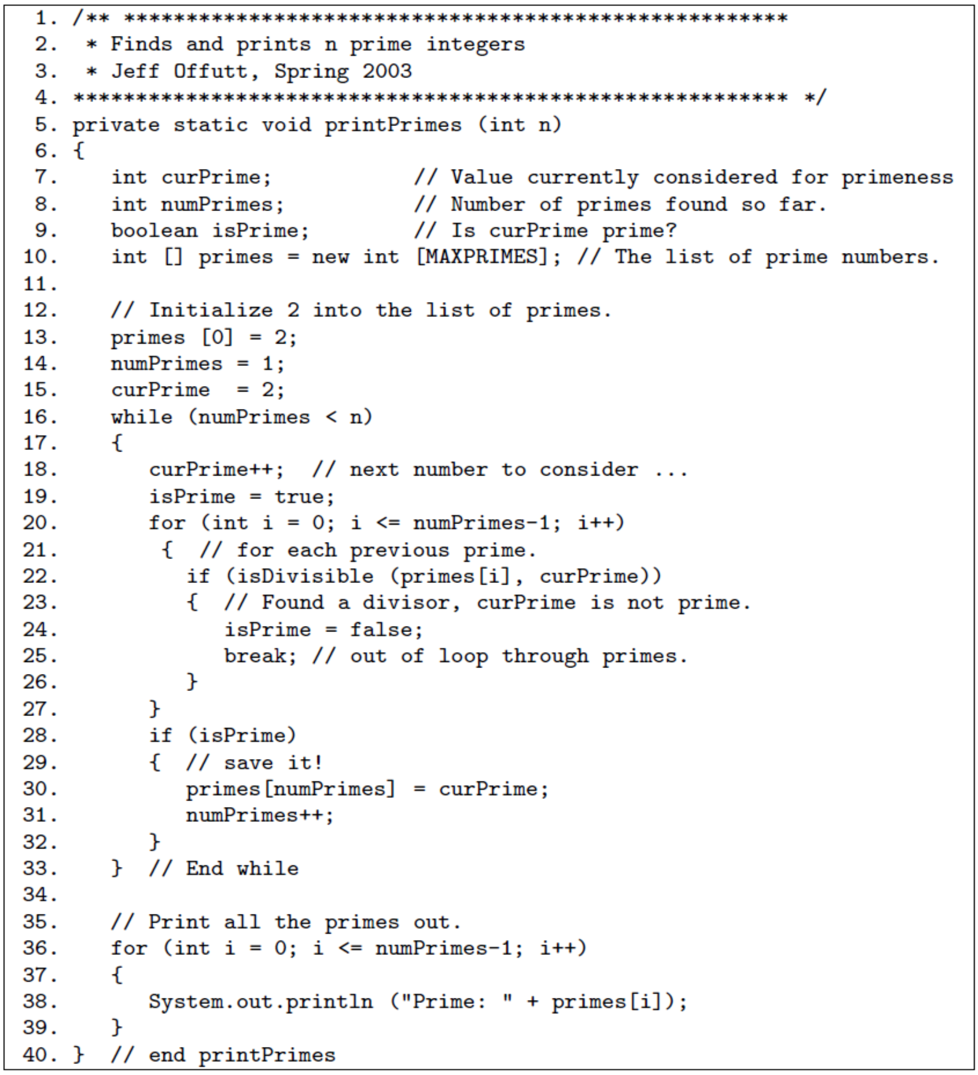
   (a) Consider test cases t1:(n = 3) and t2:(n = 5). Although these tour the same prime paths in printPrimes(), they do not necessarily find the same faults. Design a simple fault that t2 would be more likely to discover than t1 would (note that the fault should not change the control flow graph).

   (b) 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 second `for` statement without going through the body of the while loop.

   (c) Enumerate the test requirements for Node Coverage, Edge Coverage, and Prime Path Coverage for the graph for printPrimes(). Please write down the test requirements for prime path in an increasing order of a size of test requirements.

   (d) List a set of test paths that achieve Node Coverage but not Edge Coverage on the graph.

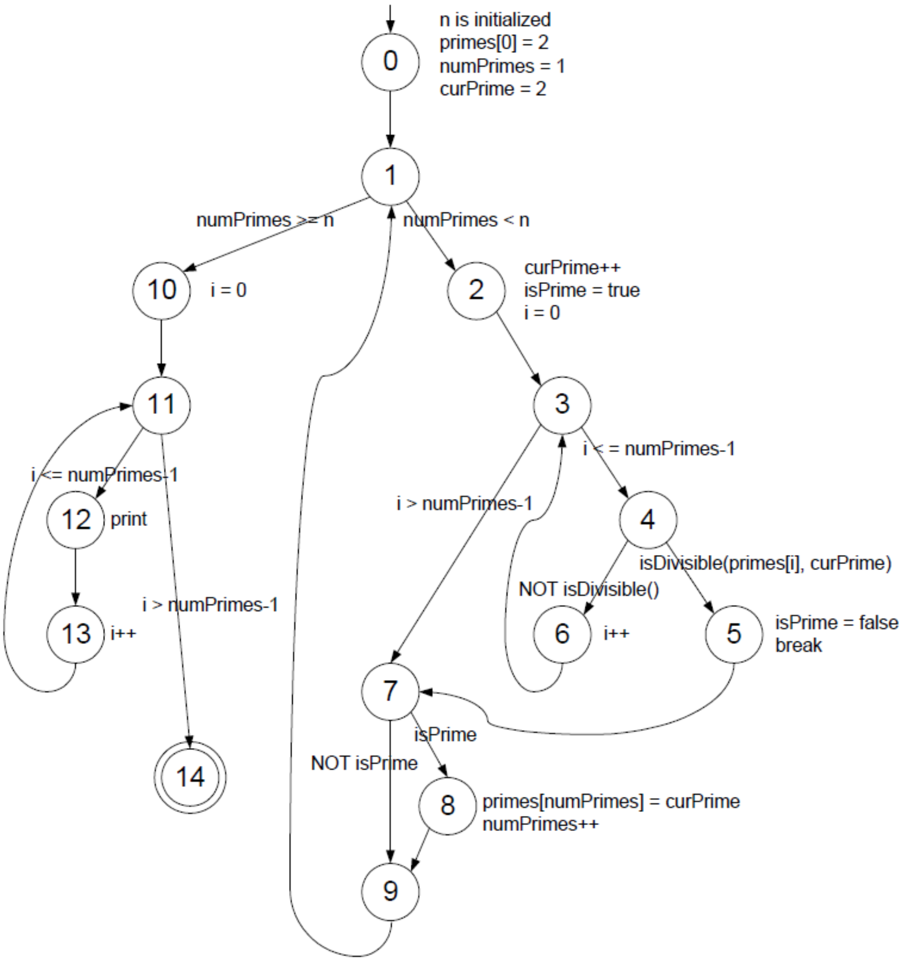
   (e) List a set of test paths that achieve Edge Coverage but not Prime Path Coverage on the graph.



```java
/** *****************************************************
 * Finds and prints n prime integers
 * Jeff Offutt, Spring 2003
********************************************************** */
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]);
   }
}  // end printPrimes
```


0
n is initialized
primes[0] = 2
numPrimes = 1
curPrime = 2
1
numPrimes >= n
numPrimes < n
10
i = 0
2
curPrime++
isPrime = true
i = 0
11
3
i <= numPrimes-1
i < = numPrimes-1
12
print
i > numPrimes-1
4
isDivisible(primes[i], curPrime)
NOT isDivisible()
13
i++
i > numPrimes-1
6
i++
5
isPrime = false
break
7
isPrime
NOT isPrime
14
8
primes[numPrimes] = curPrime
numPrimes++
9