Homework #4: Due Apr 26

1. Use the following method fmtRewrap() for questions a-e below.
   (a) Draw the control flow graph for the fmtRewrap() method.
   (b) For fmtRewrap(), find a test case such that the corresponding test path visits the edge that connects the beginning of the while statement to the S = new String(SArr) + CR; 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 fmtRewrap().
   (d) List test paths that achieve Node Coverage but not Edge Coverage on the graph.
   (e) List test paths that achieve Edge Coverage but not prime Path Coverage on the graph.

2. Use the following method printPrimes() for questions a-f below.
   (a) Draw the control flow graph for the printPrimes() method.
   (b) Consider test cases $t_1 = (n = 3)$ and $t_2 = (n = 5)$. Although these tour the same prime paths in printPrimes(), they do not necessarily find the same faults. Design a simple fault that t_2 would be more likely to discover than t_1 would.
   (c) 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 without going through the body of the while loop.
   (d) Enumerate the test requirements for Node Coverage, Edge Coverage, and Prime Path Coverage for the graph for printPrimes().
   (e) List test paths that achieve Node Coverage but not Edge Coverage on the graph.
   (f) List test paths that achieve Edge Coverage but not Prime Path Coverage on the graph.

```
 1. /** ****************************************************
 2.  *  Rewraps the string (Similar to the Unix fmt).
 3.  *  Given a string S, eliminate existing CRs and add CRs to the
 4.  *  closest spaces before column N.  Two CRs in a row are considered to
 5.  *  be "hard CRs" and are left alone.
 7. ******************************************************** */
 6.
 8. static final char CR = '\n';
 9. static final int inWord      = 0;
10. static final int betweenWord = 1;
11. static final int lineBreak   = 2;
12. static final int crFound     = 3;
13. static private String fmtRewrap (String S, int N)
14. {
15.    int state = betweenWord;
16.    int lastSpace = -1;
17.    int col = 1;
18.    int i = 0;
19.    char c;
20.
21.    char SArr [] = S.toCharArray();
22.    while (i < S.length())
23.    {
24.       c = SArr[i];
25.       col++;
26.       if (col >= N)
27.          state = lineBreak;
28.       else if (c == CR)
29.          state = crFound;
30.       else if (c == ' ')
31.          state = betweenWord;
32.       else
33.          state = inWord;
34.       switch (state)
35.       {
36.       case betweenWord:
37.          lastSpace = i;
38.          break;
39.
40.       case lineBreak:
41.          SArr [lastSpace] = CR;
42.          col = i-lastSpace;
43.          break;
44.
45.       case crFound:
46.          if (i+1 < S.length() && SArr[i+1] == CR)
47.          {
48.             i++; // Two CRs => hard return
49.             col = 1;
50.          }
51.          else
52.             SArr[i] = ' ';
53.          break;
54.
55.       case inWord:
56.       default:
57.          break;
58.       }  // end switch
59.       i++;
60.    }  // end while
61.    S = new String (SArr) + CR;
62.    return (S);
63. }
```

```
 1. /** ******************************************************
 2.  * Finds and prints n prime integers
 3.  * Jeff Offutt, Spring 2003
 4. ****************************************************** */
 5. private static void printPrimes (int n)
 6. {
 7.    int curPrime;           // Value currently considered for primeness
 8.    int numPrimes;          // Number of primes found so far.
 9.    boolean isPrime;        // Is curPrime prime?
10.    int [] primes = new int [MAXPRIMES]; // The list of prime numbers.
11.
12.    // Initialize 2 into the list of primes.
13.    primes [0] = 2;
14.    numPrimes = 1;
15.    curPrime  = 2;
16.    while (numPrimes < n)
17.    {
18.       curPrime++;  // next number to consider ...
19.       isPrime = true;
20.       for (int i = 0; i <= numPrimes-1; i++)
21.        {  // for each previous prime.
22.          if (isDivisible (primes[i], curPrime))
23.          {  // Found a divisor, curPrime is not prime.
24.             isPrime = false;
25.             break; // out of loop through primes.
26.          }
27.       }
28.       if (isPrime)
29.       {  // save it!
30.          primes[numPrimes] = curPrime;
31.          numPrimes++;
32.       }
33.    }  // End while
34.
35.    // Print all the primes out.
36.    for (int i = 0; i <= numPrimes-1; i++)
37.    {
38.       System.out.println ("Prime: " + primes[i]);
39.    }
40. }  // end printPrimes
```