

Software Testing, Quality Assurance & Maintenance (ECE453/CS447/SE465): Assignment 1

Patrick Lam

Due: February 6, 2009 (no extensions)

I have slightly modified the questions that I've taken from the book. You may discuss the assignment with others, but I expect each of you to do the assignment independently. I will follow UW's Policy 71 if I discover any cases of plagiarism.

Question 1 (5 points)

(Section 1.3, Q1): Suppose that coverage criterion C_1 subsumes coverage criterion C_2 . Also suppose that test set T_1 satisfies C_1 on program P and test set T_2 satisfies C_2 , also on P .

- (0 points) Does T_1 necessarily satisfy C_2 ? Explain, giving an example if necessary.
- (0 points) Does T_2 necessarily satisfy C_1 ? Explain, giving an example if necessary.
- (5 points) If P contains a fault, and T_2 reveals the fault, T_1 does *not* necessarily also reveal the fault. Explain.

Question 2 (15 points)

(Section 2.2.1, Q8): Design and implement a program that will compute all prime paths in a graph, then derive test paths to tour the prime paths. (Note

that p. 39 of the textbook describes how to find prime paths.) Accept graphs in graphviz format, e.g.

```
digraph G {
    a0 -> a1 -> a2 -> a3;
    b0 -> b1 -> b2 -> b3;
    start -> a0;
    start -> b0;
    a1 -> b3;
    b2 -> a3;
    a3 -> a0;
    a3 -> end;
    b3 -> end;
}
```

This format enables you to run `graphviz` to visualize your graphs. Assume that the initial node names all begin with “start” and that the final node names all begin with “end”.

Question 3 (15 points)

Test your program. Describe how you tested it and what tools you used. (This question is open-ended. Your task is to convince the marker that you did a good job in testing your program.)

Question 4 (5 points)

(Section 2.3, Q4) Consider the pattern matching example in Figure 2.21. In particular, consider the final table of tests in Section 2.3. Consider the variable $iSub$. Number the (unique) test cases, starting at 1, from the top of the $iSub$ part of the table. For example, $(ab, c, -1)$, which appears twice in the $iSub$ portion of the table, should be labelled test t_4 .

- Give a minimal test set that satisfies *all defs* coverage.
- Give a minimal test set that satisfies *all uses* coverage.
- Give a minimal test set that satisfies *all du-paths* coverage.

Question 5 (10 points)

(Section 2.3, Q7) Use the following method `printPrimes()` for questions a-f below.

```
/** *****
 * 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
}
```

- (1) Draw the control flow graph for the `printPrimes()` method.
- (3) 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.

- (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 without going through the body of the while loop.
- (1) Enumerate the test requirements for Node Coverage, Edge Coverage, and Prime Path Coverage for the graph for `printPrimes()`.
- (1) List test paths that achieve Node Coverage but not Edge Coverage on the graph.
- (1) List test paths that achieve Edge Coverage but not Prime Path Coverage on the graph.

Question 6 (5 points)

(Section 2.5, Q2) For the following questions a-c, consider the method FSM for a (simplified) programmable thermostat. Suppose the variables that define the state and the methods that transition between states are:

```

partOfDay : {Wake, Sleep}
temp      : {Low, High}

// Initially "Wake" at "Low" temperature

// Effects: Advance to next part of day
public void advance();

// Effects: Make current temp higher, if possible
public void up();

// Effects: Make current temp lower, if possible
public void down();

```

- (1) How many states are there?
- (2) Draw and label the states (with variable values) and transitions (with method names). Notice that all of the methods are total.
- (2) A test case is simply a sequence of method calls. Provide a test set that satisfies Edge Coverage on your graph.

Question 7 (30 points)

Download the Vuze file-sharing client source code (version 4.0.0.4)

<http://azureus.sourceforge.net/download.php>

and build it. You'll need to put some additional libraries in the `build/libs` subdirectory of your source tree. You can find them at

<http://www.patricklam.ca/stqam/files/a1>.

- The Vuze zipfile contains one JUnit unit test. (2) What is it? (3) Improve the coverage of the JUnit class, show evidence that you ran the modified JUnit class, and hand in your modified JUnit class (in diff form).
- (25 points) The `org.gudy.azureus2.core3.disk.impl.DiskManagerImpl` class contains a `deleteDataFiles` method that contains the TODO comment “only remove empty directories that are created for the torrent”. One way (but not the only way!) to implement this is by creating a marker file in directories that are created for the torrent, as long as you clean up the marker file eventually. Implement this TODO and hand in the diff.