

Software Testing, Quality Assurance & Maintenance (ECE453/CS447/CS647/SE465): Final

April 16, 2009

This open-book final has 7 questions. Answer 6 of the questions. Please indicate which questions you would like me to mark. All questions have equal weight. Answer the questions in your answer book. You may consult any printed material (books, notes, etc). Have fun, and best of luck after Convocation.

Question 1 (10 points): Logic Coverage

- (5) Give a Java method where CACC imposes more test requirements than PC. Explain why.
- (5) Give a predicate for which CoC does not subsume RACC.

Question 2 (10 points): Concurrency

Here is a broken concurrent program.

```
public class C {
    final List n = new LinkedList();
    public static void main(String[] argv) {
        new Thread() {
            public void run() {
                n.add(new Object());
            }
        }.start();
        n.add(new Object());
    }
}
```

- (5) Diagnose the problem. (2) How would you find the concurrency error?
(3) How would you fix it?

Question 3 (10 points): Mutation

The textbook defines a number of object-oriented mutation operators. Consider DTC, Declared Type Change:

The declared type of each new object is changed in the declaration.

The declared type of variable `v` in the following declaration is `Object`:

```
Object v;
```

- (6) Present an example of a DTC mutation, and (4) explain why DTC always produces either equivalent or stillborn mutants. (It is therefore worthless as a mutation operator!)

Question 4 (10 points): Mutation

Here is some code from `SweetHome3D`.

```
public class Room implements Serializable {
    private float [][] points;
    // etc.

    /**
     * Creates a room from its name and the given
     * coordinates.
     */
    public Room(float [][] points) {
        this.points = deepCopy(points);
        // etc.
    }

    /**
     * Returns points of the polygon matching this room.
     * @return an array of the (x,y) coordinates of the
     *         room points.
     */
    public float [][] getPoints() {
```

```

    return deepCopy(this.points);
}

private float [][] deepCopy(float [][] points) {
    float [][] pointsCopy = new float [points.length][];
    for (int i = 0; i < points.length; i++) {
        pointsCopy [i] = points [i].clone();
    }
    return pointsCopy;
}
}

```

Consider the following mutant:

```

--- Room.java 2009-04-07 18:08:01.000000000 -0400
+++ RoomMutant.java 2009-04-07 18:10:54.000000000 -0400
@@ -1,12 +1,12 @@
- public class Room implements java.io.Serializable {
+ public class RoomMutant implements java.io.Serializable {
    private float [][] points;
    // etc.

    /**
     * Creates a room from its name and the given coordinates.
     */
- public Room(float [][] points) {
-     this.points = deepCopy(points);
+ public RoomMutant(float [][] points) {
+     this.points = points;
    // etc.
}

```

(6) Show me a test harness (unit test) and, if necessary, input to the test harness that strongly kills this mutant. In other words, exhibit a sequence of calls to Room that strongly kills the mutant when you instead call RoomMutant in the sequence.

(4) How can you make RoomMutant directly substitutable for Room? You can propose changes to the code as you like.

Question 5 (10 points): Input Space Coverage

Propose a good input space partitioning for this method (use a functionality-based input domain model):

```
static boolean isPrime(int n) {
    for (int i = 2; i < (int)(Math.sqrt(n)); i++)
        if (n % i == 0)
            return true;
    return false;
}
```

(Not for credit: Find two errors in `isPrime`.)

Question 6 (10 points): Input Space Coverage/Graph Coverage

Suppose that test set T exhaustively covers the input space of a method `foo()`. Assume that `foo()` does not read any state (fields, files). Further assume that `foo()` has no unreachable code. (5) Explain why or why not T ensures prime path coverage of `foo()`. (5) Explain why or why not T ensures complete path coverage of `foo()`. (It suffices to give an example to explain why not.)

Question 7 (10 points): Graph Coverage

Consider the method `foo` and the two classes `M` and `N`. Assume there are no other implementers of `I`.

```
public static void foo(int q, I i) {
    if (q == 1)
        i = new M();
    else if (q == 2)
        i = new N();

    for (int k = 0; k < q; k++)
        System.out.println(k);
}
```

```
    i.m();
}

interface I {
    void m();
}

class M implements I {
    public void m() {
        System.out.println("m");
    }
}

class N implements I {
    public void m() {
        System.out.println("n");
    }
}
```

Enumerate the requirements for Prime Path Coverage on this example (including the two implementations of `m`). If PPC is feasible, provide a test set which achieves it. Otherwise, provide the test set that you feel comes closest to achieving PPC.