Summer School RIO 2011 Course C5

Automated Test Generation and Repair

Lab Exercise 1
Assigned: Mon, Feb 14, 2011
Available for discussion: Tue, Feb 15, 2011 3pm-5pm (email Darko to schedule a meeting)

The problems are based on some examples from Chapter 1 of the book "Introduction to Software Testing" by Ammann and Offutt.

Problem 1: (Based on Exercise **2** after Section **1.1** in the book.) The following exercise is intended to encourage you to think of testing in a more rigorous way than you may be used to. The exercise also hints at the strong relationship between specification clarity, faults, and test cases.
(a): Write a Java method **union** with the signature

public static Vector union(Vector a, Vector b)

The method should return a **Vector** of objects that are in either of the two argument **Vector** objects. Describe in code comments how you interpreted the notion of union.
(b): Upon reflection, you may discover a variety of defects and ambiguities in the given assignment. In other words, ample opportunities for faults exist. Identify at least five potential faults.
(c): Write in JUnit a set of test cases that you think would have a reasonable chance of revealing the faults you identified above. Document a rationale for each test in your test set. If possible, characterize all of your rationales in some concise summary. Run your tests against your implementation.
(d): (optional) If you're familiar with code coverage, what statement coverage do your tests achieve? You can use EclEmma or a similar tool to measure coverage, and it need not be 100%. Is 100% statement coverage feasible for your code?

Problem 2: (Based on an old exercise after Section **1.2** in the book.) Consider the following class **Count**:

```
public class Count {
   // Effects: return the number of occurrences of 0 in a
   public static int numZero(int[] a) {
      int count = 0;
      for (int i = 1; i < a.length; i++) {
         if (a[i] == 0) {
            count++;
         }
      }
      return count;
   }
   public static void main(String[] args) {
      ........
   }
}
```

(a): This program contains a fault. What is it? Does executing the program necessarily result in either incorrect output or in failure?
(b): Write a JUnit test case that results in failure. Verify by executing this test case.
(c): Write another JUnit test case that does not result in failure. Verify by executing this test case.
(d): (optional) Complete this program by modifying the **main** method to contain exactly one call to the **numZero** method. The arguments for the call can be read from the command line (parsing **args**) or from the standard input (for an example, see **http://mir.cs.illinois.edu/~marinov/sp09-cs498dm/trityp.java**).