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

Patrick Lam

Due: January 18, 2010

Question 1 (20 points)

[no solution provided]

Question 2 (10 points)

```
import org.junit.*;
import static org.junit.Assert.*;

import java.io.PrintStream;
import java.io.OutputStream;
import java.io.ByteArrayOutputStream;

/** Test class for class M, due to Xavier Noumbissi and Samaneh
 * Navabpour. */
public class TestM {

    private M m_i;
    private StringBuffer arg;
    private String myA;
    private String myB;
    private String myZero;

    private PrintStream originalOut;
    private OutputStream os;
    private PrintStream ps;
    private String separator;

    public TestM() {
        os = new ByteArrayOutputStream();
```

```
        ps = new PrintStream(os);
        separator = System.getProperty("line.separator");
        myA = "a" + separator;
        myB = "b" + separator;
        myZero = "zero" + separator;
    }

    /* Use redirection to capture output. */
    private void getOut() {
        originalOut = System.out;
        System.setOut(ps);
    }

    private void releaseOut() {
        System.setOut(originalOut);
    }

    /**
     * We have to create a new instance of M
     * for each test of the method M.m()
     */
    @Before
    public void setup() {
        m_i = new M();
        arg = new StringBuffer();
    }

    @After
    public void tearDown() {
        m_i = null;
        arg = null;
    }


    /* NODE COVERAGE AND EDGE COVERAGE*/

    @Test
    public void testM_NE_1() {
        getOut();
        m_i.m(arg.toString(), 0);

        assertEquals(myZero, os.toString());
        releaseOut();
    }

    @Test
    public void testM_NE_2() {
        getOut();
        arg.append("1");
        m_i.m(arg.toString(), 1);

        assertEquals(myA, os.toString());
        releaseOut();
    }

    @Test
    public void testM_NE_3() {
```

```
    getOut();
    arg.append("12");
    m_i.m(arg.toString(), 2);

    assertEquals(myB, os.toString());
    releaseOut();

}

@Test
public void testM_NE_4() {
    getOut();
    arg.append("123");
    m_i.m(arg.toString(), 3);

    assertEquals(myB, os.toString());
    releaseOut();
}



/*EDGE PAIR COVERAGE*/

@Test
public void testM_EP_1_0() {
    getOut();
    m_i.m(arg.toString(), 0);

    assertEquals(myZero, os.toString());
    releaseOut();
}

@Test
public void testM_EP_1_1() {
    getOut();
    m_i.m(arg.toString(), 1);

    assertEquals(myZero, os.toString());
    releaseOut();
}


@Test
public void testM_EP_2_0() {
    getOut();
    arg.append("1");
    m_i.m(arg.toString(), 0);

    assertEquals(myA, os.toString());
    releaseOut();
}

@Test
public void testM_EP_2_1() {
    getOut();
    arg.append("1");
    m_i.m(arg.toString(), 1);
```

```
        assertEquals(myA, os.toString());
        releaseOut();
    }

    @Test
    public void testM_EP_3_0() {
        getOut();
        arg.append("12");
        m_i.m(arg.toString(), 0);

        assertEquals(myB, os.toString());
        releaseOut();

    }

    @Test
    public void testM_EP_3_1() {
        getOut();
        arg.append("12");
        m_i.m(arg.toString(), 2);

        assertEquals(myB, os.toString());
        releaseOut();

    }

    @Test
    public void testM_EP_4_0() {
        getOut();
        arg.append("123");
        m_i.m(arg.toString(), 0);

        assertEquals(myB, os.toString());
        releaseOut();
    }

    @Test
    public void testM_EP_4_1() {
        getOut();
        arg.append("123");
        m_i.m(arg.toString(), 3);

        assertEquals(myB, os.toString());
        releaseOut();
    }

}
```

Question 3 (20 points)

(notes from Mehdi Amoui Kalareh)

The code for addNode() and addEdge() methods are straight forward. The only important note is that you can either write your code in a way that complies with the given test cases for candidate coverage criteria, or add additional testCases to cover criteria (like node or edge coverage) of your choice.

Part C of the question is a little trickier. Here you are asked to implement an algorithm to traverse the programs CFG. You can implement any tree search algorithms like depth-first-search or breath-first- search. However, as you are dealing with graphs rather than trees, you should keep track of the visited nodes to prevent infinite loops in your code. The implementation can be done in either a recursive or non-recursive style. You can also use any data structure you like to implement the stack of nodes, e.g. arrays.

The last part of the question is dependent on your implementation for the prior parts of the question. However, for most implementations the given test cases are not adequate to cover the isReachable() for node or edge coverage: the provided class `M` does not cover all the possible cases. Consider a case in which the start and the end nodes are the same.