SYSC 4101

Laboratory 11

During this lab you will apply a peculiar test construction technique which can be sometimes counter intuitive because you will be creating faults during testing.

Consider the code of the `OrdSetSimple` class and more specifically the `difference()` method.

```
1  public OrdSetSimple difference(OrdSetSimple s2) {
2      OrdSetSimple s1 = this;
3      OrdSetSimple newSet=new OrdSetSimple(s2.getSize());
4      for(int k=0; k<s1.getSize(); k++)
5              if (s2.find(s1.getElementAt(k)) < 0)
6                    newSet.addElement(s1.getElementAt(k));
7      return newSet;
8    }
9
```

You will repeatedly create a faulty version of this code and try to reveal each created fault with some well-crafted test(s), thereby incrementally construct a test suite.
The general procedure is the following:

1. Create a fault version of the code, only one fault at a time.
2. Design a test case that can reveal the fault and add this test to the test suite. This means you identify an input, an expected result and include the oracle in your test driver.
3. Repeat from step 1 until you have considered you created enough faulty versions.

More specifically, for the `difference()` method, follow the steps below (faults to consider are highlighted).

0. Initially your test suite TS is empty; you do not have any test case.
1. Suppose a fault has been made on line 2.
   Instead of `OrdSetSimple s1 = this`
   a developer has written
   `OrdSetSimple s1 = s2`.
   a. Create this fault, i.e., modify the original code (above) accordingly.
   b. Design and execute a test case that reveals this fault and add it to TS.
2. Suppose a fault has been made on line 2.
   Instead of `OrdSetSimple s1 = this`
   a developer has written
   `OrdSetSimple s1 = new OrdSetSimple(s2.getSize())`.
   a. Create this fault, i.e., modify the original code (above) accordingly.
   b. Execute all the tests in your test suite TS (there should only be one test at this time) to see whether any test reveals this fault.
   c. If the fault has not been revealed by the entire TS, i.e., by any of its test cases, design and execute a new test case that reveals this fault and add it to TS.

3. Suppose a fault has been made on line 2.
   Instead of `OrdSetSimple s1 = this`
   a developer has written
   `OrdSetSimple s1 = new OrdSetSimple(this.getSize()).`
   a. Create this fault, i.e., modify the original code (above) accordingly.
   b. Execute all the tests in TS to see whether any test case reveals this fault.
   c. If the fault has not been revealed by the entire TS, design and execute a new test case that reveals this fault and add it to TS.
4. Suppose a fault has been made on line 4.
   Instead of `for(int k=0; k<s1.getSize(); k++)`
   a developer has written
   `for(int k=1; k<s1.getSize(); k++)`
   a. Create this fault, i.e., modify the original code (above) accordingly.
   b. Execute all the tests in TS to see whether any test case reveals this fault.
   c. If the fault has not been revealed by the entire TS, design and execute a new test case that reveals this fault and add it to TS.
5. Suppose a fault has been made on line 4.
   Instead of `for(int k=0; k<s1.getSize(); k++)`
   a developer has written
   `for(int k=0; k<=s1.getSize(); k++)`
   a. Create this fault, i.e., modify the original code (above) accordingly.
   b. Execute all the tests in TS to see whether any test case reveals this fault.
   c. If the fault has not been revealed by the entire TS, design and execute a new test case that reveals this fault and add it to TS.
6. Suppose a fault has been made on line 4.
   Instead of `for(int k=0; k<s1.getSize(); k++)`
   a developer has written
   `for(int k=0; k<s1.getSize(); k--)`
   a. Create this fault, i.e., modify the original code (above) accordingly.
   b. Execute all the tests in TS to see whether any test case reveals this fault.
   c. If the fault has not been revealed by the entire TS, design and execute a new test case that reveals this fault and add it to TS.
7. Suppose a fault has been made on line 4.
   Instead of `for(int k=0; k<s1.getSize(); k++)`
   a developer has written
   `for(int k=0; k<this.getSize(); k++)`
   … you know the drill now …
8. Suppose a fault has been made on line 4.
   Instead of `for(int k=0; k<s1.getSize(); k++)`
   a developer has written
   `for(int k=0; k<s2.getSize(); k++)`
   … you know the drill now …
9. Suppose a fault has been made on line 4.
   Instead of `for(int k=0; k<s1.getSize(); k++)`
   a developer has written
   `for(int k=0; k<s1.getCapacity(); k++)`
   … you know the drill now …

10. Suppose a fault has been made on line 5.
    Instead of `s2.find(s1.getElementAt(k)) < 0`
    a developer has written
    `s2.find(s1.getElementAt(k+1)) < 0`
    … you know the drill now …
11. Suppose a fault has been made on line 5.
    Instead of `s2.find(s1.getElementAt(k)) < 0`
    a developer has written
    `s2.find(s1.getElementAt(k)) <= 0`
    … you know the drill now …
12. Suppose a fault has been made on line 5.
    Instead of `s2.find(s1.getElementAt(k)) < 0`
    a developer has written
    `s2.find(s1.getElementAt(k-1)) < 0`
    … you know the drill now …
13. Suppose a fault has been made on line 5.
    Instead of `s2.find(s1.getElementAt(k)) < 0`
    a developer has written
    `s2.find(s2.getElementAt(k)) < 0`
    … you know the drill now …
14. Suppose a fault has been made on line 5.
    Instead of `s2.find(s1.getElementAt(k)) < 0`
    a developer has written
    `s2.find(s1.getElementAt(k)) > 0`
    … you know the drill now …
15. Suppose a fault has been made on line 6.
    Instead of `newSet.addElement(s1.getElementAt(k))`
    a developer has written
    `newSet.addElement(s1.getElementAt(k+1))`
    … you know the drill now …
16. Suppose a fault has been made on line 6.
    Instead of `newSet.addElement(s1.getElementAt(k))`
    a developer has written
    `newSet.addElement(s1.getElementAt(k-1))`
    … you know the drill now …
17. Suppose a fault has been made on line 6.
    Instead of `newSet.addElement(s1.getElementAt(k))`
    a developer has written
    `newSet.addElement(s2.getElementAt(k))`
    … you know the drill now …
18. Suppose a fault has been made on line 6.
    Instead of `newSet.addElement(s1.getElementAt(k))`
    a developer has written
    `s2.addElement(s1.getElementAt(k))`
    … you know the drill now …

19. Suppose a fault has been made on line 3.
    Instead of `OrdSetSimple newSet=new OrdSetSimple(s2.getSize())`
    a developer has written
    `OrdSetSimple newSet=new OrdSetSimple(s1.getSize())`.

What do you conclude, especially from step 19?