Ciências ULisboa | Informática

VVS 2018/2019
Exam 2 – June 26, 2019
Duration: 3 hours

Consider the following SUT, a method from a class implementing a splay tree. A splay tree is a self-adjusting binary tree that change after each operation, bringing the current key to the root (so that the next access of x is faster). The following method brings parameter `key` to the tree's root.

```
1  private void splay(E key) {  // E implements comparable
2    Node<E> l, r, t, y, header = new Node<E>();
3    l = r = header;
4    t = root;
5    header.left = header.right = null;
6    for (;;) {
7      if (key.compareTo(t.key) < 0) {
8        if (t.left == null) break;
9        if (key.compareTo(t.left.key) < 0) {
10         y = t.left;           // rotate right
11         t.left = y.right;
12         y.right = t;
13         t = y;
14         if (t.left == null) break;
15       }
16       r.left = t;             // link right
17       r = t;
18       t = t.left;
19     } else if (key.compareTo(t.key) > 0) {
20       if (t.right == null) break;
21       if (key.compareTo(t.right.key) > 0) {
22         y = t.right;          // rotate left
23         t.right = y.left;
24         y.left = t;
25         t = y;
26         if (t.right == null) break;
27       }
28       l.right = t;            // link left
29       l = t;
30       t = t.right;
31     } else break;
32   } // for
33   l.right = t.left;
34   r.left = t.right;
35   t.left = header.right;
36   t.right = header.left;
37   root = t;
38 }
```

Group 1

1. Draw the control flow graph (CFG) of method `splay()`. Assume method calls as simple commands.
2. List the requirements produced by edge-pair coverage (do not need to include simple node and edge requirements).
3. List the prime paths of the CFG that are cycles.

Group 2

1. Identify the definitions and usages of variables `r`, `l` and `header`.
2. Write a table with the DU paths of the previous variables.
3. Describe the definition of *All Defs Coverage* and *All DU Path Coverage* criteria. What would be the number of requirements for both criteria considering variables `r`, `l` and `header`?

Group 3

Consider the following code:

```
1  public boolean m(int a, int b, int c) {
2    if (a > 0 && (b < 100 || c < 100)) {
3      int d = b + c;
4      if (d > 100)
5        d = b-c;
6      else
7        d = 0;
8    }
9    return a < b || d > 0;
10 }
```

1. Find the reachability logic formulas for each predicate.
2. Find the determination predicates for each clause of the predicates of method `m()`.
3. Using *Clause Coverage*, list its requirements.
4. Write JUnit tests to cover these requirements. If there are infeasible requirements, justify why is it so.

Group 4

1. Using QuickCheck, create a random string generator, `StringGen`, only with vogals and digits, and having length 10.

   Consider method `count(String s)` that sums all digits from a given string. For example, `count("12ae8")==11`. Let's call this result the *value* of a string.
2. Implement a property that states that the value of a string `s` is half the value of string `s+s`.
3. Implement another property that states that the sum of values of string `s1` and `s2` is equal to the value of the concatenation of both strings by any order.
4. Why it is important for any fake data random generation to closely follow the real data distribution?