Ciências ULisboa | Informática

VVS 2017/2018
Exam 2 – June 27, 2018
Duration: 3 hours

Consider the class implementing an iterator for a `LinkedBinaryTree`, using linked node objects. Class `Node`, not shown here, includes attributes `left` and `right` to reference the left and right sub-trees, and attribute `data` for the respective element. The iterator uses an auxiliary stack to iterate over all tree elements.

```
1   private class SuffixIterator implements Iterator<E> {
2     private ArrayStack<Node<E>> stack;
3
4     public SuffixIterator() {
5       stack = new ArrayStack<Node<E>>();
6       Node<E> current = root;
7       // push on nodes from root to
8       // leftmost descendant
9       while (current != null) {
10        stack.push(current);
11        if (current.left != null)
12          current = current.left;
13        else
14          current = current.right;
15      }
16    }
17
18    public boolean hasNext() {
19      return !stack.isEmpty();
20    }
21
22    public E next() {
23      if (stack.isEmpty())
24        throw new NoSuchElementException();
25
26      Node<E> current = stack.peek();
27      stack.pop();
28      Node<E> node = current;
29      if (!stack.isEmpty()){
30        Node<E> parent = stack.peek();
31        if (current == parent.left) {
32          current = parent.right;
33          while (current != null){
34            stack.push(current);
35            if (current.left != null)
36              current = current.left;
37            else
38              current = current.right;
39          }
40        }
41      }
42      return node.data;
43    }
44  }
```

Group 1

1. Draw the control flow graph (CFG) of method `next()`. Assume the calls of `stack()` methods as simple commands.
2. List the requirements produced by branch coverage of `next()`. List each requirement as an edge of the CFG.
3. Write a set of JUnit tests that satisfy the previous requirements. To produce these tests consider that `LinkedBinaryTree<E>` includes two constructors, a nullary one for making empty trees, and a ternary one that receives a left tree, an information (consider integers), and a right tree. The class returns an iterator using method `iterator()`.
4. Identify the prime paths of the CFG.

Group 2

1. What is the meaning of saying that coverage criteria A subsumes coverage criteria B?
2. Describe the following Input Space Partitioning criteria: Each Choice Coverage (ECC), Pairwise Coverage (PWC), Base Choice Coverage (BCC) and All Combinations Coverage (ACoC). Use a small example to show which requirements each criteria would produce. Also, describe all subsumption relations between these criteria.

Group 3

Consider the following code:

```
1   /**  Check if a given string is a palindrome.
2    *   A palindrome is a string that is the same
3    *   when read right-to-left.
4    */
5   public static boolean isPalindrome(String s) {
6     if (s == null)
7       throw new NullPointerException();
8     int left = 0;
9     int right = s.length() - 1;
10    boolean result = true;
11    while (left < right && result == true) {
12      if (s.charAt(left) != s.charAt(right))
13        result = false;
14      left++;
15      right--;
16    }
17    return result;
18  }
```

1. Using *Clause Coverage*, list its requirements.
2. Write JUnit tests to test these requirements. If there are infeasible requirements, justify why is it so
3. Compute the determination predicates for each predicate of method `isPalindrome()`.

Group 4

Consider that you have access to a string random generator `StringGen`, in the context of JUnit QuickCheck, that includes only lowercase letters.

1. Implement the property that for any string `s`, string `s+reverse(s)` is a palindrome. Assume access to method `String reverse(String s)`.
2. What is this property's body describing?

```
1    assumeTrue(isPalindrome(s));
2    assertTrue("...", isPalindrome(s+s));
```

3. What is the problem with the previous code? Implement a property that solves it without using a palindrome generator.
4. Implement a palindrome generator and refactor the previous test.