Online Test COMP2100/6442 Semester 2

Question 1:
This question is composed of two items:

- A Binary Search Tree Question (item 1.a) and
- A Branch Complete Question (item 1.b).

Please upload the following files to Wattle after you complete the tasks: `BST.java` and `BSTBranchCompleteTest.java`. In each file **write your UID and name where indicated** in the code.

1.a) [4 marks] A Binary Search Tree (BST) is used to **generate a sequence of bases** (A, G, C, T) in a DNA molecule. The sequence is generated based on the information stored in each node (Character **base**, see node representation) and the following rules:

1. All bases of all nodes that have an even number of direct children under the key node (inclusive) are concatenated;
2. If the given key is not found, return "TCAG";
3. If the given key is the root node and it has no children, return **null**.

****Note that 0 is considered an even number.****

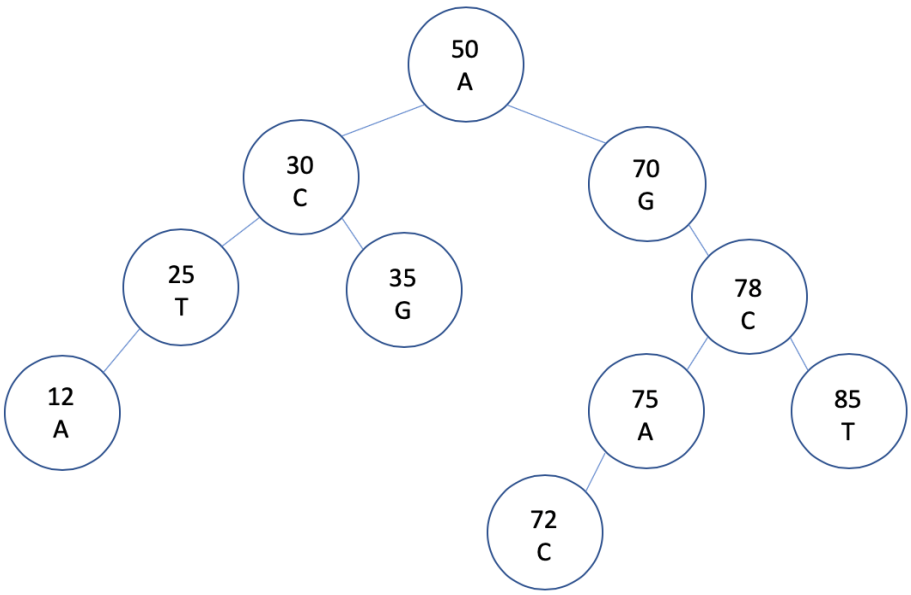
Note that the order in which the nodes are concatenated is irrelevant. As long as the rules are met, the sequence of bases generated will be considered valid.

You can define additional methods in the BST class to complete the task.

The method signature (the method that will be tested) is:

public String DNAGenerator(int key)

Example:



Example 1:
Key: 25
Result: A
In this example only the node 12-A has a direct even number of children under the key node 25-T (rule 1).

Example 2:
Key: 30
Result: ACG (or any sequence of bases valid [AGC, GCA, GAC, ...])
In this example the nodes 12-A, 30-C (key node inclusive), and 35-G have a direct even number of children (rule 1). Note that 25-T is not included as it has an odd number of direct children.

Example 3:
Key: 17
Result: TCAG
In this example the key node is not found (rule 2).

Example 4:
Key: 50
Result: ACGACCT (or any sequence of bases valid [ACGACCT, ACGACTC, ...])
In this example the nodes 12-A, 30-C, 35-G, 50-A (key node inclusive), 72-C, 78-C and 85-T have a direct even number of children (rule 1). All other nodes are not included as they have an odd number of direct children.

1.b) [2 marks] Implement the minimum number of JUnit test cases for **DNATreeCalc()** that is branch complete. Read the code of **DNATreeCalc()** in the `BranchComplete.java` file and implement the **minimum number of test cases** in the `BSTBranchCompleteTest.java` file.

Note that each execution of an assertion counts as a single test case, therefore loops that execute the same assertion multiple times count as multiple tests.