Homework 7

Question 1 (5 pt.)

The height of a node in a tree can be recursively defined as the maximum height of its child nodes plus 1. The height of a leaf node is 0. The height of a tree can be defined as the height of its root node.

a) (3 pt.) Copy files `Node.java` and `BinarySearchTree.java` written in class into a new directory named `q1`. Add the following functions to the binary search tree implementation:

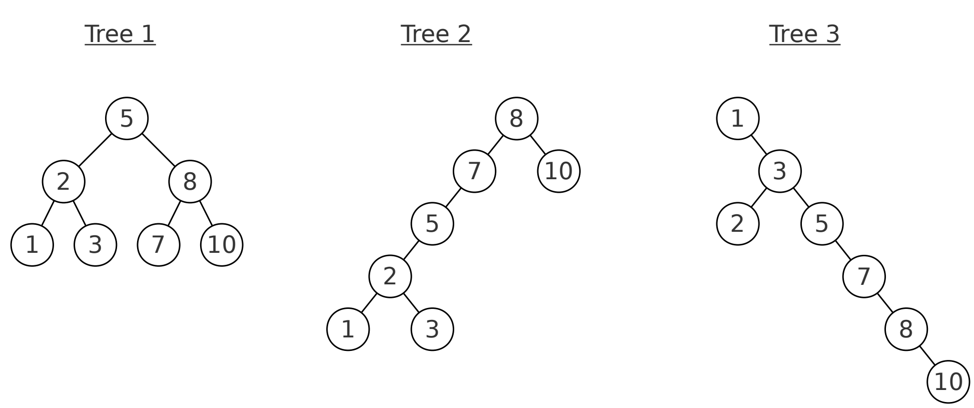
- A private function `int GetNodeHeight(Node node)` that returns the height of a node. This function should use a recursive implementation that mirrors the definition presented above.

- A public function `int GetHeight(Comparable key)` that returns the height of the node containing the given key, or –1 if the key is not present in the tree.

- An overloaded public function `int GetHeight()` that returns the height of the tree, or –1 if the tree is empty.

b) (2 pt.) Write a main program in a file named `Test.java`. This program should instantiate three binary search trees (`tree1`, `tree2`, and `tree3`) containing keys 1, 2, 3, 5, 7, 8, and 10, and leaving the data values set to `null`. For each tree, insert its elements in the appropriate order to reach the distinct topologies shown below.

The main program should print the height of these trees by invoking function `GetHeight()`. Run your program and verify that the height values are as expected.



Create a ZIP file called `q1.zip` with the full content of directory `q1`, and upload it on Canvas.

Question 2 (5 pt.)

Copy files `Node.java` and `BinarySearchTree.java` written in class into a new directory named `q2`.

a) (3 pt.) Write a main program in a file named `Test.java`. The aim of this program is evaluating the performance of the binary search tree under different conditions. The program expects two arguments being passed from the shell command line:

   - The first argument is a string set to "`sorted`" or "`random`". If set to "`sorted`", your program inserts integer values into the binary search tree in order, from lowest to highest. If set to "`random`", your program inserts random integer values into the tree in any order. The data associated with the inserted keys can be set to `null` in both cases.

   - The second argument is an integer representing the number of values to insert into the binary search tree.

   The program should implement the following additional features:

   - If it is invoked with the wrong number of arguments, it should print an error message and exit.

   - If the first argument is not set to "`sorted`" or "`random`", it should print an error message and exit.

   - The insertion operation should be protected with a `try`-`catch` clause in order to prevent occasional duplicate key errors when inserting random numbers.

   - Once all requested values have been inserted in the tree, the program finishes without providing any output.

   Create a ZIP file named `q2.zip` containing directory `q2` and upload it on Canvas. Your program should compile correctly and run without errors.

b) (2 pt.) Pick 5 linearly increasing values for the number of elements (e.g., 1000, 2000, …, 5000) in such a way that they lead to execution times somewhere between 0.1 and 10 seconds (short enough, yet distinguishable from noise). Do this separately for the *sorted* and *random* insertion policies.

Run your program with the `time` command, and use the first time component displayed by the command's output. This is a sample execution:

```
$ time java Test sorted 10000
real    0m0.906s
user    0m1.785s
sys     0m0.056s
```

Describe the observed results. Your answer should include a description of the theoretical asymptotic cost of your program, and how it is reflected in the results. Upload a file named `q2.pdf` on Canvas containing the collected data and the written answer.