**Home project #2**

**Due to October 21, 11:59 pm. No late submissions are accepted!**

Your program code should start with a comment with your name in it.

Your program should be submitted ONLY through the Blackboard. You should submit only the source code (.cpp and header files).

The assignment is for the individual work only. You are responsible for every statement in your code.

In this assignment we implement containers based on BST and AVL trees and compare their performance. Also, we practice using basic inheritance.

You should submit me few files: header and implementation files for containers (maybe separate library for ach container, maybe one library for both), driver file with experiments in it and a text file with experiments results.

1. Implement a class template ***BST<class Type>*** that stores items of class Type in form of a binary search tree. Objects of class Type are guaranteed to be comparable and have operator<<. There should be the following methods:

* Default constructor that initializes empty container.
* Destructor. Make sure destructor avoids memory leaks.
* Height() function returns the height of the tree.
* Size() function returns the number of items in container.
* Print() function that prints the content of the container sorted in ascending order.
* Subscripting operator (operator[]) that returns a reference to the item.
* Insert( const Type & ).
* Remove( const Type & ).
* Copy assignment operator.
* Also, you can add to it any number of private utility functions if you want.

1. Implement a class template ***AVL<class Type>*** that stores items of class Type in form of a AVL tree. Objects of class Type are guaranteed to be comparable and have operator<<. There should be the same public methods as in case of BST and any other private methods that you need. Class AVL should be derived from BST.

1. Conduct a series of experiments to measure the performance of each container. Each experiment of size **N** goes like this:

before the experiment:

* Create a vector of **N** random integers.

the experiment itself:

* Create an empty container of integers and Insert() all integers from the vector in it one by one.
* Remove each second integer in vector from container.

Measure the duration of the experiment itself. Measurement should be done using standard *chrono* library.

Fill the following table with experiment durations in milliseconds:

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | 1000 | 100’000 | 1’000’000 |
| BST |  |  |  |
| AVL |  |  |  |

          This table should be submitted in a text file with your program.

Additional:

* Feel free to use the same ***Node<Type>***class template for both BST and AVL. It means that Node can include height information even for BST.
* Feel free to use standard C++ smart pointers. It is not mandatory but will be clean and pro. Also, it simplifies things like destructor.

Good luck!