Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Fall 2020-2021

Homework #3

Due: 01/12/2020 - 23:55

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You HAVE TO write down the code on your own.
You CANNOT HELP any friend while coding.
Plagiarism will not be tolerated!!

1 Introduction

In computer science, trees are very important to store, process and query data efficiently. We already talked about binary search trees (BST) that are used to search a specific element inside a set. The aim of this homework is to implement a comprehensive class of `BinarySearchTree`.

The `BinaryTree` class has a private root of the binary search tree (`TreeNode* root`). As always, when traversed in in-order notation, the tree elements are sorted. Some of the overloaded operators will be part of the class itself, while others will be free one, thus not as a member of the `BinaryTree` class *(Hint: Pay attention to `main.cpp` to determine which functions are going to be free).* You also need to implement **(deep) copy constructor** and **destructor** for `BinaryTree`.

Apart from these small implementations, there is an `Iterator` class, which is located at `BinaryTree.h` and `BinaryTree.cpp`. This class should work with `BinaryTree` class. There are some TODOs in this file. Every TODO part that you have to implement is marked clearly in the source code. Also, a `Stack` class is given for using in the `Iterators` class. You do not need to change anything on `Stack.h` and `Stack.cpp`.

In addtition, a `main.cpp` provided and it uses these structures. You do not need to change anything particular to this one, but please keep in mind that we will use various different test files to grade your homework. Therefore, it is recommended that you try your implementations with handmade test files as well. Please make sure that your program gives reasonable results especially with the `main.cpp` file. The output is not going to be the same as provided below (i.e Sample Run) since insertion is done with random elements, however you still need to pay attention to what is changing between operators.

2 Operators

The operators to be implemented are listed below in detail.

<<	this operator takes an instance of `BinaryTree` class and prints its content in in-order notation by sending it to an output (ostream) stream.
=	this operator will assign (more accurately make replica-clone of) the right hand side tree to the left one. In order to allow multiple assignments (e.g. tree1=tree2=tree3) you should carefully pay attention to this operator.
==	takes an instance of tree as parameter. The operator should return true if and only if both contains the same elements. This means that although the construction of two trees (i.e. insert order) is different, this function should return true.
!=	negation of the above operator, i.e. returns false if trees contains same elements.
+=	takes a tree as parameter and adds its nodes to the current tree instance. Note that this operation should alter the nodes of current tree.
+	takes a tree instance as parameters and combines content of the current one and the one on the parameter. After the operation, content of the current tree object should not be altered, instead, another instance of a tree should be returned.
+=	takes an integer as parameter and adds a node containing that integer to the tree. Note that this operation should alter the nodes of current tree.
+	takes a tree and an integer **(either as a left hand side operand or a right hand side!)** and returns another instance where the number is added to the tree. (Yes, there are more than one version of + and += operator. They differ in terms of function signatures)

3 Sample Runs

Below is the output of main.cpp file. Notice that rand is used, so do not pay attention to exact number, instead pay attention what has changed between two operators.

```
Duplicate value found in tree.
//tree: in-order print with Iterator class
16 22 36 50 78 84 87 93 94

//tree: in-order print with << operator
16 22 36 50 78 84 87 93 94

//tree_2 (copied from tree):
16 22 36 50 78 84 87 93 94

//tree_2 += 124
tree_2: 16 22 36 50 78 84 87 93 94 124

//tree_3 = tree_2 + 245 + 987 + 457
tree_3: 16 22 36 50 78 84 87 93 94 124 245 457 987

//Random number generation for tree_4 and tree_5
//tree_4 += (rand() % 100 + 1)
//tree_5 = (rand() % 100 + 50) + tree_5
(1. iteration) tree_4: 63
(1. iteration) tree_5: 77
(2. iteration) tree_4: 63 91
(2. iteration) tree_5: 77 109
(3. iteration) tree_4: 63 64 91
(3. iteration) tree_5: 76 77 109
Duplicate value found in tree.
(4. iteration) tree_4: 41 63 64 91
(4. iteration) tree_5: 76 77 109
(5. iteration) tree_4: 41 63 64 73 91
(5. iteration) tree_5: 76 77 86 109
```

```
//tree_6 += 25
tree_6 (before): 25

Content of tree_4 and tree_5 are not the same
rand() works like a charm
//tree_6 = tree_4 + tree_5

tree_6 (after) : 41 63 64 73 76 77 86 91 109
```

4 Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homeworks we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases.**

What and where to submit (PLEASE READ, IMPORTANT): You should prepare (or at least test) your program using MS Visual Studio 2012 or 2019 C++. We will use the standard C++ compiler and libraries of the above mentioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course). Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.
Name your cpp file that contains your program as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.cpp

Your SUCourse user name is actually your SUNet username that is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the folder name must be:

cago_Ozbugsizkodyazaroglu_Caglayan_hw1.cpp

Do not add any other character or phrase to the folder name. Make sure that it contains the last version of your homework program. Compress this folder using WINZIP or WINRAR program. Please use "zip" compression. **"rar" or another compression mechanism is NOT allowed.**. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your cpp file.
You will receive no credits if your compressed folder does not expand or it does not contain the correct files. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names. **Submit via SUCourse ONLY!** You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!

CS204 Team (Fatih Taşyaran, Kamer Kaya)