Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Fall 2020-2021

Homework #4

Due: 12/12/2020 - 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

Templates are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one. We already talked about and implemented binary search trees (BST) that are used to search a specific element inside a set. The aim of this homework is to make `BinarySearchTree` templated.

Fully implemented `BinaryTree`, `Iterator` and `Stack` classes are provided for this homework and are the same with Homework#3. However, there is a new driver code in `main.cpp` and your templated classes should work perfectly with this new `main.cpp`. 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. Moreover, you could use your own implementations from HW3 or start with provided solution and **the structure of the files may change while templating the classes. You may submit less files than provided solution.** You could also make changes in `Iterator` and `Stack` classes.

2 Sample Runs

```
//————————————————————//
int tree:
//————————————————————//

//tree: in-order print with Iterator class
19 22 82 99 105 114 127 156 170 182

//tree: in-order print with << operator
19 22 82 99 105 114 127  156 170 182

//tree_2 (copied from tree):
19 22 82 99 105 114 127  156 170 182

Duplicate value found in tree.
//tree_2 += arr[24]
```

tree_2: 19 22 82 99 105 114 127 156 170 182

Duplicate value found in tree.
//tree_3 = tree_2 + arr[45] + arr[87] + arr[57]
tree_3: 19 22 65 82 99 105 114 127 156 165 170 182

//Random access for tree_4 and tree_5
//tree_4 += (arr[(rand()+1)%100])
//tree_5 = (arr[(rand()+50)%100]) + tree_5
(1. iteration) tree_4: 82
(1. iteration) tree_5: 74
(2. iteration) tree_4: 82 168
(2. iteration) tree_5: 46 74
(3. iteration) tree_4: 32 82 168
(3. iteration) tree_5: 46 74 165
(4. iteration) tree_4: 32 82 156 168
(4. iteration) tree_5: 32 46 74 165
(5. iteration) tree_4: 32 82 127 156 168
(5. iteration) tree_5: 32 46 74 156 165

//tree_6 += arr[25]
tree_6 (before): 167

Content of tree_4 and tree_5 are not the same
rand() works like a charm
//tree_6 = tree_4 + tree_5
Duplicate value found in tree.
Duplicate value found in tree.

tree_6 (after) : 32 46 74 82 127 156 165 168

//————————————————//
float tree:
//————————————————//

Duplicate value found in tree.
//tree: in-order print with Iterator class
5.40576 13.7232 22.3656 39.4383 63.5712 76.0249 76.823 89.1529 97.0634

//tree: in-order print with << operator
5.40576 13.7232 22.3656 39.4383 63.5712 76.0249 76.823 89.1529 97.0634

//tree_2 (copied from tree):
5.40576 13.7232 22.3656 39.4383 63.5712 76.0249 76.823 89.1529 97.0634

//tree_2 += arr[24]
tree_2: 5.40576 13.7232 22.3656 39.4383 63.5712 76.0249 76.823 89.1529 93.1835 97.0634
//tree_3 = tree_2 + arr[45] + arr[87] + arr[57] tree_3: 0.416161 5.40576 12.5475 13.7232 22.3656 39.4383 63.5712 65.313 76.0249 76.823 89.1529 93.1835 97.0634

//Random access for tree_4 and tree_5
//tree_4 += (arr[(rand()+1)%100])
//tree_5 = (arr[(rand()+50)%100]) + tree_5
(1. iteration) tree_4: 12.6075
(1. iteration) tree_5: 68.7387
(2. iteration) tree_4: 12.6075 98.4752
(2. iteration) tree_5: 19.2214 68.7387
(3. iteration) tree_4: 12.6075 60.6969 98.4752
(3. iteration) tree_5: 0.323146 19.2214 68.7387
(4. iteration) tree_4: 12.6075 33.0337 60.6969 98.4752
(4. iteration) tree_5: 0.323146 19.2214 39.4383 68.7387
(5. iteration) tree_4: 12.6075 33.0337 60.6969 73.0729 98.4752
(5. iteration) tree_5: 0.323146 19.2214 39.4383 40.0944 68.7387

//tree_6 += arr[25]
tree_6 (before): 28.4293

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) : 0.323146 12.6075 19.2214 33.0337 39.4383 40.0944 60.6969 68.7387 73.0729 98.4752

//————————————————//
double tree
//————————————————//

//tree: in-order print with Iterator class
0.357857 3.92803 32.8374 41.3984 53.976 62.948 65.1132 73.8534 77.4273 92.397

//tree: in-order print with << operator
0.357857 3.92803 32.8374 41.3984 53.976 62.948 65.1132 73.8534 77.4273 92.397

//tree_2 (copied from tree):
0.357857 3.92803 32.8374 41.3984 53.976 62.948 65.1132 73.8534 77.4273 92.397

//tree_2 += arr[24]
tree_2: 0.357857 3.92803 32.8374 41.3984 53.976 62.948 65.1132 73.8534 77.4273 92.397 93.081

//tree_3 = tree_2 + arr[45] + arr[87] + arr[57]
tree_3: 0.357857 3.92803 25.7265 32.8374 41.3984 53.976 62.948 65.1132 73.8534 77.4273 79.347 92.3069 92.397 93.081

//Random access for tree_4 and tree_5
//tree_4 += (arr[(rand()+1)%100])
//tree_5 = (arr[(rand()+50)%100]) + tree_5
(1. iteration) tree_4: 65.1132
(1. iteration) tree_5: 78.3099
(2. iteration) tree_4: 65.1132 68.7861
(2. iteration) tree_5: 43.1953 78.3099
(3. iteration) tree_4: 3.92803 65.1132 68.7861
(3. iteration) tree_5: 43.1953 78.3099 96.7405
(4. iteration) tree_4: 3.92803 65.1132 68.7861 93.5004
(4. iteration) tree_5: 32.9642 43.1953 78.3099 96.7405
(5. iteration) tree_4: 3.92803 54.8042 65.1132 68.7861 93.5004
(5. iteration) tree_5: 32.9642 43.1953 78.3099 87.7614 96.7405

//tree_6 += arr[25]
tree_6 (before): 73.8534

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) : 3.92803 32.9642 43.1953 54.8042 65.1132 68.7861 78.3099 87.7614 93.5004 96.7405

//————————————————————//
char tree
//————————————————————//

Duplicate value found in tree.
//tree: in-order print with Iterator class
, 6 ; C G Z [c t

//tree: in-order print with << operator
, 6 ; C G Z [c t

//tree_2 (copied from tree):
, 6 ; C G Z [c t

//tree_2 += arr[24]
tree_2: , 6 : ; C G Z [c t

Duplicate value found in tree.
//tree_3 = tree_2 + arr[45] + arr[87] + arr[57]
tree_3: , 6 : ; C G O Z [c t y

/Random access for tree_4 and tree_5
//tree_4 += (arr[(rand()+1)%100]) //tree_5 = (arr[(rand()+50)%100]) + tree_5
(1. iteration) tree_4: _
(1. iteration) tree_5: X
(2. iteration) tree_4: _ r
(2. iteration) tree_5: 5 X
(3. iteration) tree_4: _ l r
(3. iteration) tree_5: 5 < X
(4. iteration) tree_4: U _ l r
(4. iteration) tree_5: ’ 5 < X
Duplicate value found in tree.
(5. iteration) tree_4: U _ l r
**(5. iteration) tree_5: ’ 5 < X **

//tree_6 += arr[25]
tree_6 (before): ;

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) : ’ 5 ¡ U X _ l r

//————————————————————//
string tree
//————————————————————//

//tree: in-order print with Iterator class
a against battle galaxy secret sinister space stars title to

//tree: in-order print with << operator
a against battle galaxy secret sinister space stars title to

//tree_2 (copied from tree):
a against battle galaxy secret sinister space stars title to

//tree_2 += arr[24]

tree_2: a against battle drums galaxy secret sinister space stars title to

//tree_3 = tree_2 + arr[45] + arr[87] + arr[57]
tree_3: a against an battle drums evil galaxy secret sinister space stars striking title to

//Random access for tree_4 and tree_5
//tree_4 += (arr[(rand()+1)%100])
//tree_5 = (arr[(rand()+50)%100]) + tree_5
(1. iteration) tree_4: the
(1. iteration) tree_5: away
(2. iteration) tree_4: striking the
(2. iteration) tree_5: away their
(3. iteration) tree_4: ago striking the
(3. iteration) tree_5: away the their
Duplicate value found in tree.
(4. iteration) tree_4: ago striking the to
(4. iteration) tree_5: away the their
(5. iteration) tree_4: Rebel ago striking the to
(5. iteration) tree_5: away backdrop the their

//tree_6 += arr[25]
tree_6 (before): echo

Content of tree_4 and tree_5 are not the same
rand() works like a charm
//tree_6 = tree_4 + tree_5
Duplicate value found in tree.

tree_6 (after) : Rebel ago away backdrop striking the their to

3 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.
Compress your source file into a zip and name your zip file that contains your program as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

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.zip

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 source files.

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)