CMSC 3613

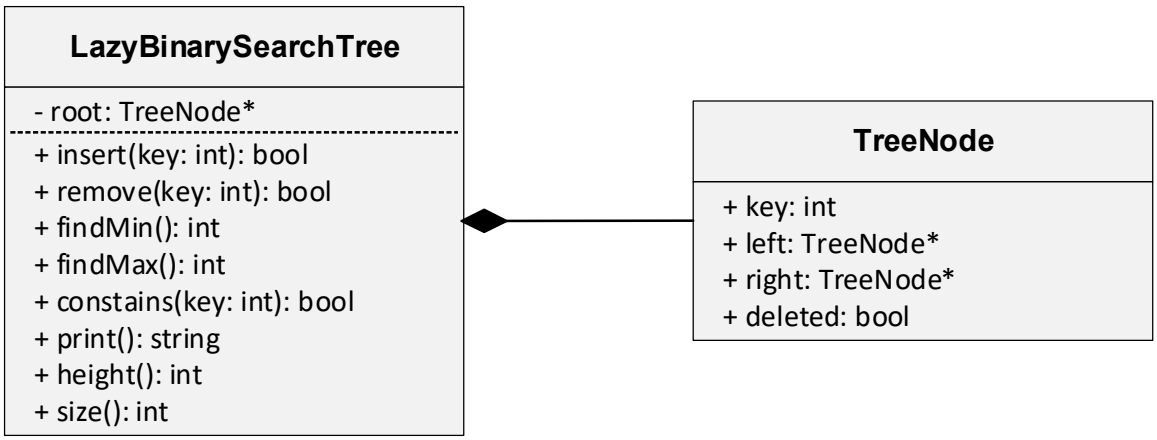
Programming Assignment: Binary Search Tree

Due Date: Check the D2L calendar for the due date.

Assignment:

The task of this project is to implement a binary search tree with lazy deletion in C++. We can assume the data type of the elements is int.

The class structures are specified by the following UML class diagram. Please follow the specification to name your classes, identifiers and functions. You may also add helper functions and additional fields as you see fit.



`insert` should insert a new element as a leaf node. The valid set of keys is all integers in the range [1,99]. If the new element would be a duplicate of a non-deleted element already in the tree, then `insert` should do nothing. However, if the new element is not a duplicate of a non-deleted element, but is a duplicate of a deleted element, then insert should "undelete" the deleted element in-place rather than physically inserting a new copy of the element. The return value of insert should indicate whether `insert` logically (as opposed to physically) inserted a new element.

`remove` should not physically remove an element from the tree. Rather, it should mark the specified element as logically deleted. If the specified element is not in the tree or is already marked as deleted, then `remove` should do nothing. The return value of `remove` should indicate whether `remove` logically deleted an element.

`findMin` should return the value of the minimum non-deleted element, or -1 if none exists.

`findMax` should return the value of the maximum non-deleted element, or -1 if none exists.

`contains` should return whether the given element both exists in the tree and is non-deleted.

`print` should perform a pre-order traversal of the tree and print the value of each element, including elements marked as deleted. However, elements that are marked as deleted should be preceded by a single asterisk (*). Every pair of adjacent elements should be separated by whitespace in the printing, but no whitespace should occur between an asterisk and the element

with which it is associated. Leading and trailing whitespace is tolerable, but it will be ignored. (no additional messages should be printed, either). An example of the output is as follows:

```
45 30 2 *5 47 50 *60
```

`height` should return the height of the tree, including "deleted" elements.

`size` should return the count of elements in the tree, including "deleted" ones.

The valid set of keys is all integers in the range [1,99]. Every function that accepts a key argument should show an appropriate message (e.g., illegal argument: value not in range) if and only if the argument is invalid.

Along with the LazyBinarySeacrTree class, please also prepare another source file that has a main function. You should name this file as p03.cpp, and follow existing examples to create the makefile, through which the name of the executable should be p03. Please **note**: this time we don't have generic types, so in your makefile you may need to build an object file for each of your LazyBinarySeacrTree class and TreeNode class.

This main function will take two command line arguments. The first argument will be the input file name and second will be output file name. The input file will be given to the program and the output file will be generated by the program. This main function will create an instance of LazyBinarySearchTree and do the operations specified in the input file.

The format of the input file (e.g. input.dat) will be similar as:

```
insert:98
insert:67
insert:55
insert:45
print
remove:84
remove:45
contains:45
findMin
findMax
print
height
size
insert:84
insert:32
insert:32
print
findMin
insert:980
insert
hiha
```

{insert and remove will be followed by ":" semicolon and the key to be inserted/removed. You have to check for validity of the key as well as the line in the file. }

The corresponding correct output file (e.g., output.dat) will be similar as:

```
true
true
true
true
98 67 55 45
false
true
false
55
98
98 67 55 *45
3
4
true
true
false
98 67 55 *45 32 84
32
Error: insert (illegal argument: not in range)
Error: insert (no key)
Error: hiha (invalid command)
```

Requirements:

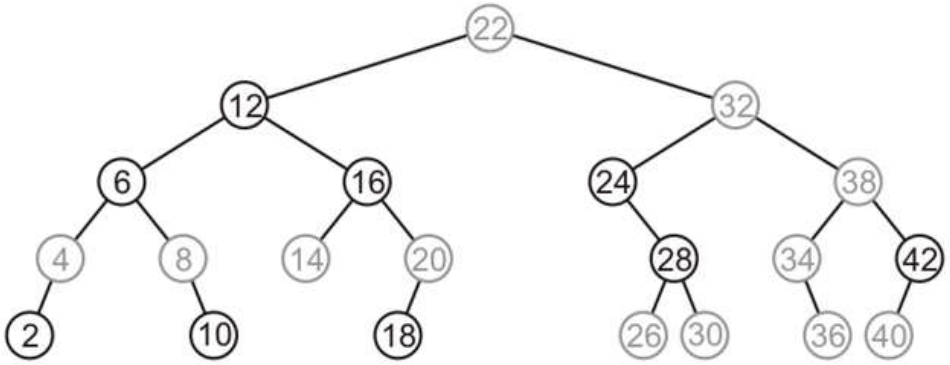
1. Your program should follow the instructions described above in the "Assignment" section. Pay attention to the underscored parts.
2. Use C++ language for implementation. Your implement should produce similar output demonstrated above (output.dat) based on the input (similar as input.dat).
3. The input and output files should be provided in command line argument as follows:
   p03 input.dat output.dat
   p03 should be the name of your executable (p03 should be fixed), and input.dat, output.dat are the names of the input file and output file (the input/output names can be any).

Evaluation:

This project will be evaluated according to the correctness of the various tree functions specified above, and secondarily according to the quality of your code. The rubric is as follows:

Categories	Weights
`insert`	10%
`remove`	5%
`findMax`	15%
`findMin`	15%
`contains`	10%
`print`	10%
`height`	10%
`size`	10%
`error checking`	5%
`report`	10%

Note: findMax() and findMin() could be complicated and their weights are bigger than other functions. Please think thoroughly. Another case for your consideration (gray nodes are "deleted"):



Submission:

1. Please provide a readme file, to help with the compilation and execution of your code. For example, information about your operating system and detailed command to compile your code should be included.

2. You need to submit a report, which should include the following items:
   1) Your name and UCO email address
   2) The project number, i.e., p03
   3) A brief discussion of your implementation: just like an explanation of your idea in an interview.
   4) A screenshot of a test run.
   5) Also copy all your source code on D2L (not image, but text contents).

Notes:

1. To be considered on time, the program must be turned in by the due date.
2. Programs must reflect your knowledge and work, not others. You may ask others questions about algorithms, methods and programming style, but when you start writing code, you must work only with your group member(s).
3. Program grades reflect both the performance of the program and the programming style of the program.