EE 441 HOMEWORK #3

Binary Search Trees

Due: December 24, 2018, 23:59
For questions: ccakmak@metu.edu.tr
hdoga@metu.edu.tr

Part 1:

You are required to design a `BinSTree` class that allows you to declare objects of type `BinSTree <T>`. The class implements a binary search tree and provides the interface to a storage of items of type `T` by hiding the details of the pointer operations.

The declaration and partial implementation of the `BinSTree` class is in `"bstree.h"`, which includes the `Node` class in the given `"treenode.h"`.

Implement all member functions except for `void BinSTree<T>::Delete(const T& item)` declared according to the comments and the visual examples below. Some of the member functions are implemented for you. Please check them carefully to see how to update the member variables to maintain the state of the `BinSTree` object.

The comments for your implementations are meant to guide you. If you have a correctly working implementation of your own it is also ok.

The `Insert` method takes the data of the node as argument, dynamically create the node using `GetTreeNode` and insert the node in the binary search tree. Insertion is always as a leaf node.

The `FindNode` method returns the node address of the searched data item and a pointer to its parent, and `NULL` otherwise. If there are multiple, it returns the address of the one found first.

Assume class `T` has all comparison operators defined.

All methods should maintain the state of the `BinSTree` object by correctly modifying the class members.

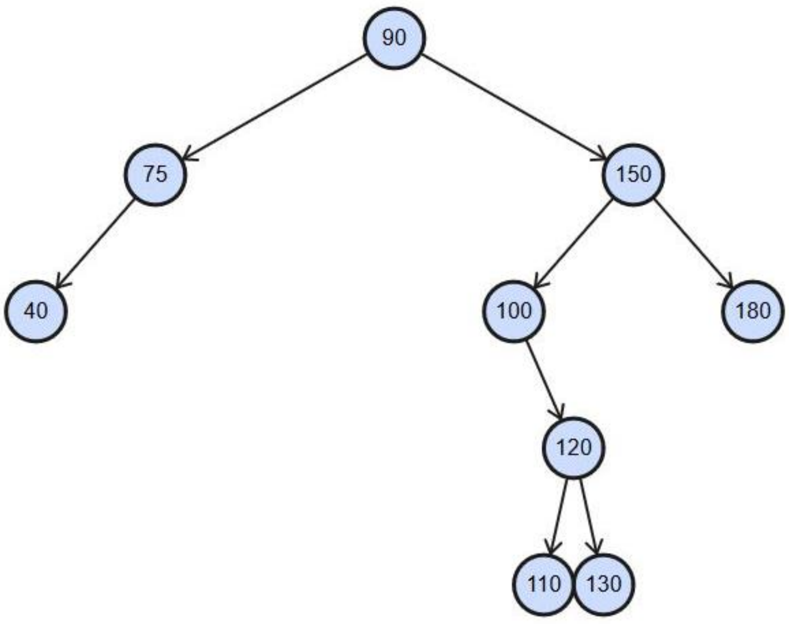
You can reuse member functions of the class (such as `CopyTree` and `DeleteTree`) to implement other member class member functions.

Please make sure that you make the necessary checks to prevent any undesired state of the `BinSTree` objects.

Note that `BinSTree` is a class template. The examples below are for a `BinSTree <int>` class. Your code will be checked using other template parameters (i.e. for different `T`s, not necessarily primitive data types) as well.

Examples:

```
BinSTree<int> MyTree;

TreeNode<int> * TreeP1, * TreeP2;

MyTree.Insert(90);

MyTree.Insert(150);

MyTree.Insert(180);

MyTree.Insert(75);

MyTree.Insert(100);

MyTree.Insert(120);

MyTree.Insert(130);

MyTree.Insert(110);

MyTree.Insert(40);
```



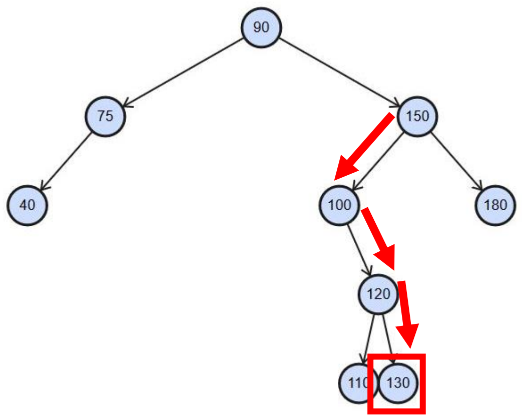
```
TreeP1=MyTree.FindNode(120,TreeP2);

cout<<TreeP2->data;
```

Output: 100

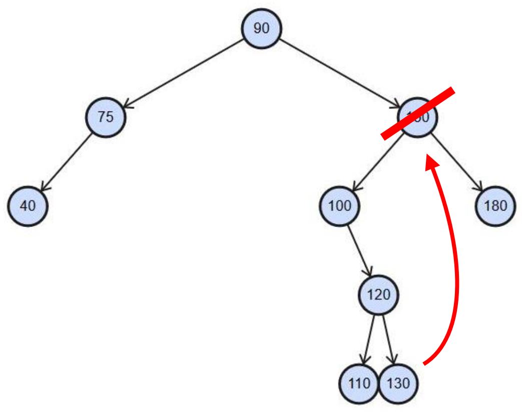
Part 2:

`void BinSTree<T>::Delete(const T& item)` declared according to the comments and the visual examples below.

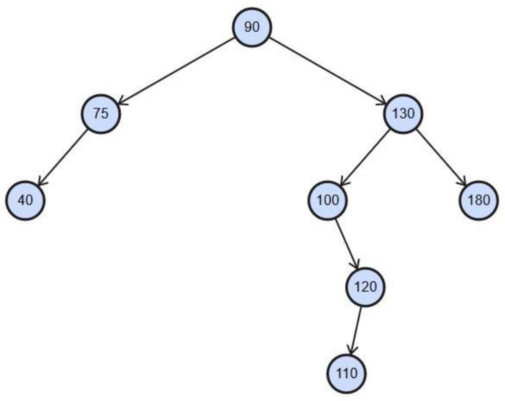
```
MyTree.Delete(150);
```



First the item to delete (150) is found. Then the item for replacement is found (The highest value on the left branch if there is any, or the lowest value on the right branch).



Then the replacement is performed. Please note that this is not a simple value change since these are not just values, they are objects indeed. Thus, the old object must be deleted and all the pointers regarding the old and the new objects (parents, children) must be updated.



The final structure of the tree.

Regulations:

1. Use **Code::Blocks IDE** and choose GNU GCC Compiler while creating your project. Name your project as “e<student_ID>_HW3”. Send the whole project folder compressed in a rar or zip file. You will not get full credit if you fail to submit your project folder as required.

2. The code you uploaded should be compilable and the built file should be executable. **We have to see some output to grade your homeworks.** So please make sure that you have uploaded a working version for your homework.

3. You should insert comments to your source code at appropriate places without including any unnecessary detail. Comments will be graded (On the condition that you have managed to send a working code). A code with insufficient/excessive comments is not a proper piece of work.

4. The homework is to be prepared individually. It is not allowed to prepare the homework as a group. METU honor code is essential. Do not share your code. **Any kind of involvement in cheating will result in a zero grade for all homeworks, for both givers and receivers.**

5. You have to give the links to any website you have benefitted from. You can add them to your code as comments in the beginning.

6. Late submissions are welcome, but penalized according to the following policy:

   - 1 day late submission: HW will be evaluated out of 70.
   - 2 days late submission: HW will be evaluated out of 50.
   - 3 days late submission: HW will be evaluated out of 30.
   - 4 or more days late submission: HW will not be evaluated.

Good Luck!