CS6140: Advanced Programming Lab
Quiz 1

September 11, 2014

1 You are given...

You are given the following C++ source files which implement a Binary Search Tree.

- main.cpp
- Node.h
- Node.cpp
- BST.h
- BST.cpp
- functions.cpp
- a few sample input files.

In addition, you are provided a working makefile. First ensure that you are able to compile the files. In particular, executing "make main" from the command line should compile all the files and create an executable file called "main".

There is a node class and a BST class, both of which are implementations of a node and binary search tree that we saw in the lab exercises. Most of the functions are already implemented for you. Some functions are not implemented and they are in functions.cpp. The file "main.cpp" holds the `main(...)` function. Familiarize yourself with the code structure. Note in particular that the program takes two filenames as input — each of the input file contains lines of the form:

jobId priority duration

The input is exactly as we have used earlier in the lab exercises. You can additionally assume that any BST will contain at most 100 nodes.

2 You are required to ...

You are required to edit ONLY "functions.cpp" and possibly main.cpp for testing purposes. Do not edit other files!!! Your final submission will include only functions.cpp

There are four functions that are left empty in "functions.cpp" that you are required to implement.

isEqualNode This function takes two Node objects and checks if the data contained in the two nodes is exactly equal. Note that, the pointers contained in the nodes need not be equal. The function must return 1 if the two nodes are equal, and 0 otherwise. Implementing this correctly will fetch you 2 marks.

isEqualBST Two trees T_1 and T_2 are equal if they

1. have the same structure, and
2. their corresponding nodes are equal.

The function must return 1 if the two trees are equal, and 0 otherwise. Implementing this correctly will fetch you 4 marks.

isEquivBST Two trees T_1 and T_2 are equivalent if, for each node present in T_1, there is an equal node present in T_2, and vice versa. You may assume that no two nodes in the same tree will be equal. You must return 1 if the two trees are equivalent. Otherwise, return 0. Implementing this correctly will fetch you 7 marks provided that your algorithm runs in $O(n)$ time.

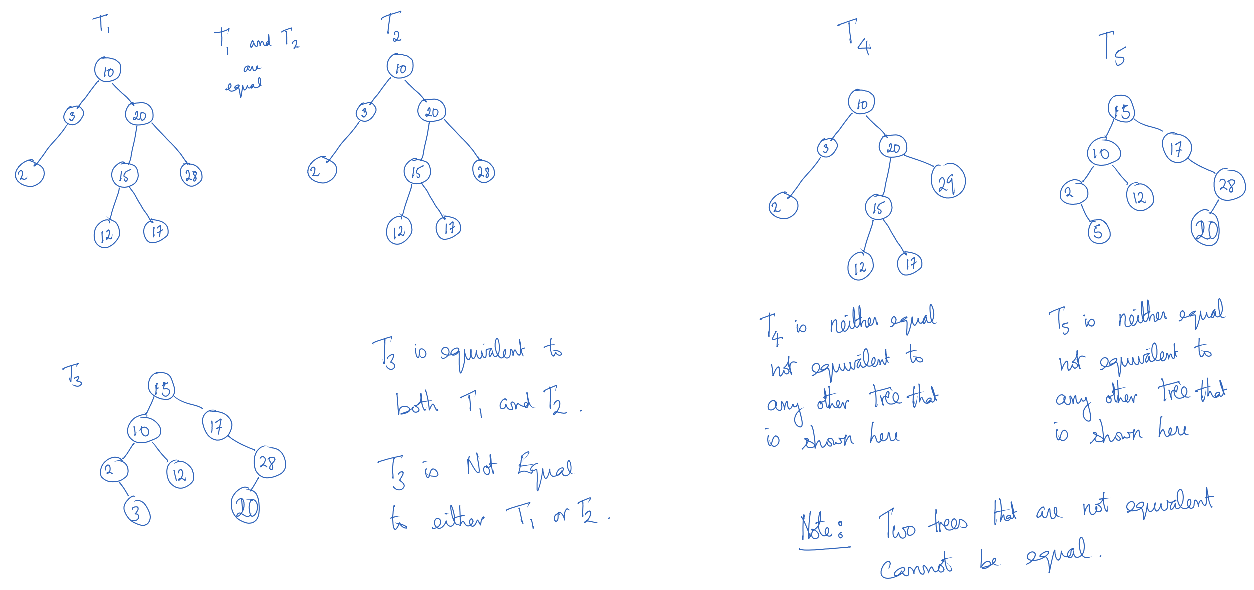
T1
T1 and T2 are equal
T2
T4
T5
T3
T3 is equivalent to both T1 and T2.
T3 is Not Equal to either T1 or T2.
T4 is neither equal nor equivalent to any other tree that is shown here
T5 is neither equal nor equivalent to any other tree that is shown here
Note: Two trees that are not equivalent cannot be equal.

Figure 1: Example trees

Fig. 1 gives examples of equal and equivalent trees. Note that the nodes in the example trees show only integers as keys. A node in our tree, however, will have jobId, priority, and duration as the data.

POPrint This function must traverse the nodes and print them in pre-order traversal in exactly the same order as in RecPOPrint(...) already provided to you. However, there is a catch. Your code must be iterative in nature and must neither use recursion nor stacks. Implementing this will fetch you 2 marks. This function will require a bit of time and has the least marks/effort ratio. So please make sure all other functions are implemented without any bugs before you invest time in this function.

You are free to implement auxiliary functions required in functions.cpp – but DO NOT edit class definitions.

3 Submission

You are required to submit your "functions.cpp" file, but it must be renamed "⟨rollnumber⟩.cpp" before you submit it. Please ensure that the file compiles without any error before you submit it.

4 Hints with Penalty

In the case that you are absolutely stuck with the design of algorithm for either of **isEqualBST** or **isEquivBST**, it is possible to ask for hints. These hints will be useful only in the design of the algorithm and there will not be any additional code supplied to you. In case you decide to take hints you will lose 2 marks per hint that you take. No hints are provided for the non-recursive POPrint.