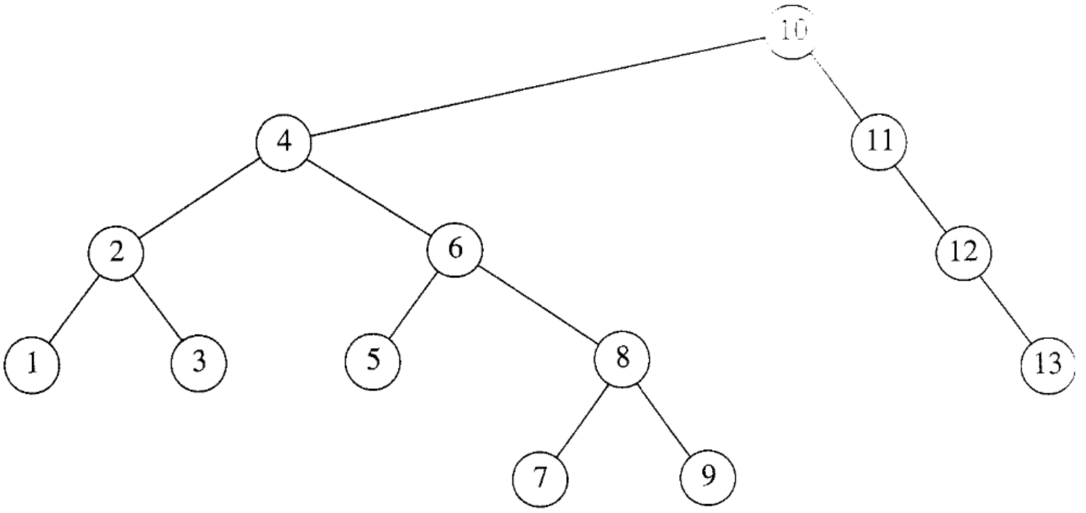
EE2331 Homework 4

Out: Nov. 10, 2023
Due: 11:59PM, Nov. 20th, 2023. No late submission will be counted/graded.

Full mark: 40 pts

1. (10 pts) Given a node in a BST, it is sometimes important to be able to find its successor in the sorted order determined by an in-order tree walk. If all keys are distinct, the successor of node x is the node with the smallest key greater than key[x]. The structure of a BST allows us to determine the successor of a node easily. Describe a procedure that returns the successor of a node x in a binary search tree if it exists, and NIL if x has the largest key in the tree. (Hint: in order to consider all cases, find each node's successor in the tree shown on the last page. Then generalize your observations.)

Note: you **are not allowed to do** an in-order traversal to get the successor of node x.



3. (30 pts) This is a programming task that showcases one of the most important operations for many data structures: **search**. Essentially, you need to implement a simple interaction system for querying/maintaining a small-scale student database.

1) I provided a database file that contains many student records. The entries are: student ID, major, grade (percentage), admission type (4-year, ASI, ASII), and admission year. Your program should ask the user to input the file name rather than hardcoding it. We will use two different input files (with the same format) to test the system.

2) Create an indexing structure for this database using **binary search tree**. Note that you cannot use any tree-related classes from STL. You should define the struct of the node and implement it from scratch (need left and right pointers). You can borrow codes from the provided BST-class-preview.cpp.

3) Ask users to input one of three operations once you finish creating the index.

1. search
2. insert
3. over

If the user inputs "search", ask: "What is the student ID", read the ID and output either the student's record or "This ID does not exist".

If the user inputs "insert", ask: "What is the record", read the whole line and check whether it has the same format as the input file. If yes, add it to the BST, otherwise output "wrong format".

If the user inputs “over”, exist the system (i.e., the program).

Field types:
Student ID (8 digits), major (a string containing at most 3 letters), admission type (4-year, ASI, ASII), and year (4 digits).

Besides BST-class-preview.cpp, you can find Q2-gen-record.cpp, which is used to generate student record file. You can use that file to generate more test inputs.