Middle East Technical University | Department of Computer Engineering

CENG 242

Programing Language Concepts

Spring 2011-2012

Homework 1

Due date: 18 March 2012, Sunday, 23:55

1 Objective

In this homework, you will be given a binary search tree (BST) and asked to retrieve a node while making the node the root node without changing the BST property. The point is to retrieve nodes in such a way that subsequent retrieve operation of the same node will be faster. In order to accomplish this task, we will use some heuristics to move the asked node to the root node.

Each node will be represented by an Integer. You should return the resulting BST with the retrieved node at the root. The movement of the retrieved node must be achieved according to the rules specified below.

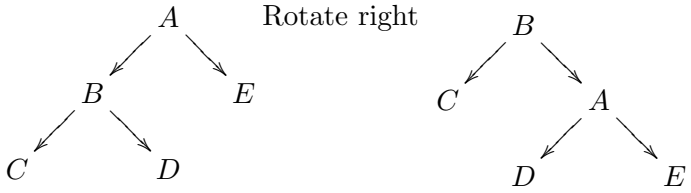
Note that the BST has the property that for every node the left subtree contains the nodes smaller than that node and the right subtree contains the nodes that are larger.

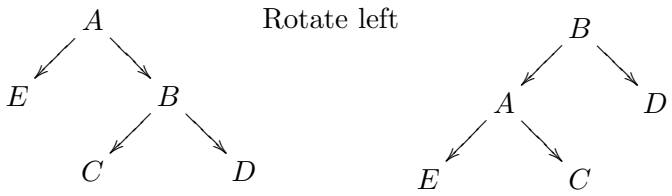
2 Specifications

Let's say that you are asked to retrieve (i.e. move to root) the node indicated by `B`. Depending on the position of `B` in the tree, different operations are need to be applied.

1. `B` **is the root:** In this case, you should simply return the same tree as `B` is already at the root.

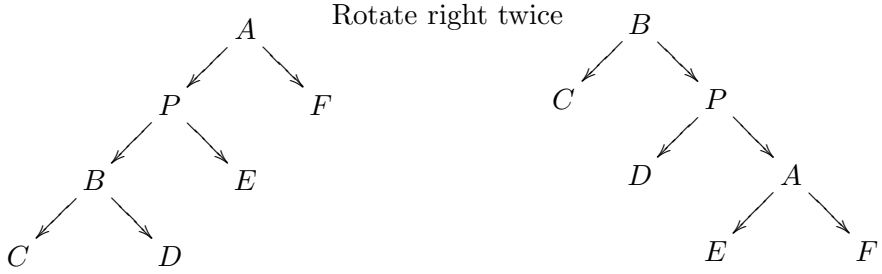
2. `B`**'s parent is the root:** In this case, you need to rotate the tree on the root node once to get `B` to the root. If `B` is the left child, you need to rotate right; if `B` is the right child you need to rotate left. How to do these rotations are shown below:



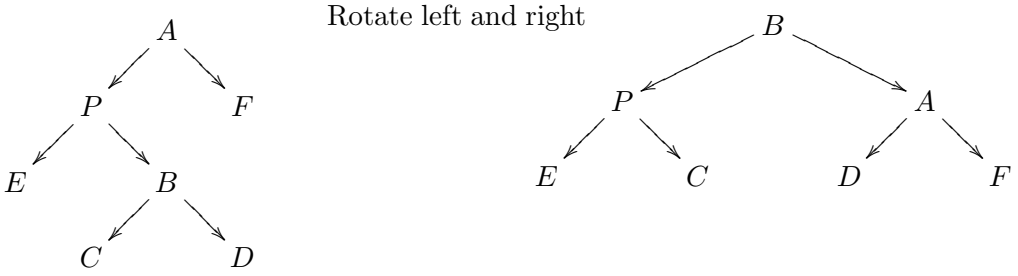


3. `B`**'s grandparent is the root:** This case needs to be examined under two sub-cases:

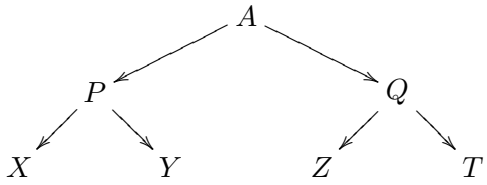
   (a) **Both** `B` **and its parent are left or right children:** In this case, you need to rotate the tree on the root node once to get `B`'s parent to the root, and then rotate again on the new root node to get `B` to the root. Below an example is shown for the left children case; the right children case is synonymous (you need to rotate left twice):



   (b) `B` **and its parent are opposite side children:** This case can happen if `B` is the left child and its parent is the right child or vice versa. In this case, you need to rotate the tree on `B`'s parent node once (left rotate if `B` is the right child and right rotate if `B` is the left child), and then rotate on the root node (again choosing left and right similarly).



4. `B` **is deeper in the tree:** In this case, you need to recursively move `B` to the grandchild position. This means, you need to identify the sub-tree that `B` should be in (using the BST property) and bring `B` to the root of that tree. Once you do that, `B` will be the grandchild of the initial root, and then you can apply the previous rule to bring it to the root. For instance, if the initial tree is as given below, you need to determine whether `B` belongs to the sub-tree denoted by `X`, `Y`, `Z`, or `T`. Then you need to recursively apply the rules given above (i.e. make a recursive function call) to make `B` the root of that subtree. Finally, you can apply the previous rule to make `B` the root.



Assume that the tree is defined as:

data Tree = Empty | Branch Integer Tree Tree deriving Show

You will write a function moveToRoot in the form:

moveToRoot :: Tree -> Integer -> Tree

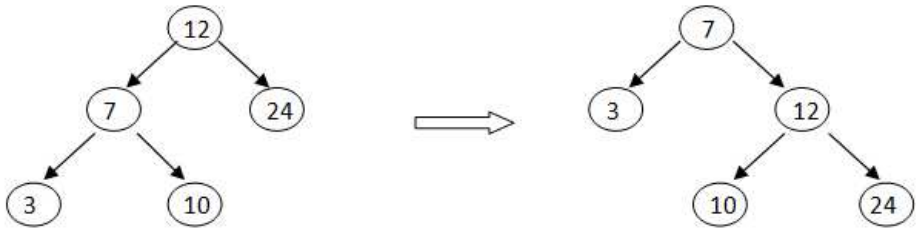
Note that;

- The nodes of the given tree will just include integers.
- The tree will be in the binary search tree form.
- We don't have a Leaf constructor in the Tree definition, the Leaf node is represented as Branch Integer Empty Empty.
- Your homework file should include both tree definition and function type annotation as they are given above.
- *moveToRoot* function isn't going to be called with a node which doesn't exist in the given tree.

3 Examples

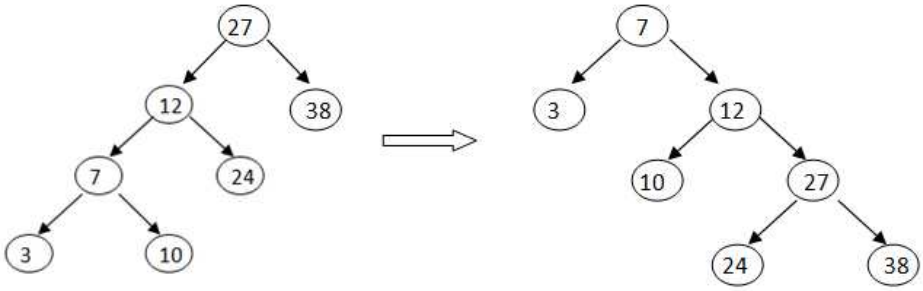
1. `moveToRoot (Branch 12 (Branch 7 (Branch 3 Empty Empty) (Branch 10 Empty Empty)) (Branch 24 Empty Empty)) 7`

   `Branch 7 (Branch 3 Empty Empty) (Branch 12 (Branch 10 Empty Empty) (Branch 24 Empty Empty))`



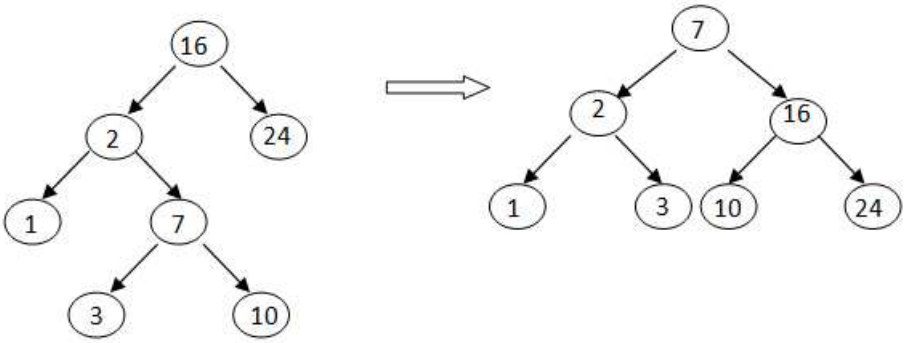
2. `moveToRoot (Branch 27 (Branch 12 (Branch 7 (Branch 3 Empty Empty) (Branch 10 Empty Empty)) (Branch 24 Empty Empty)) (Branch 38 Empty Empty)) 7`

   `Branch 7 (Branch 3 Empty Empty) (Branch 12 (Branch 10 Empty Empty) (Branch 27 (Branch 24 Empty Empty) (Branch 38 Empty Empty)))`



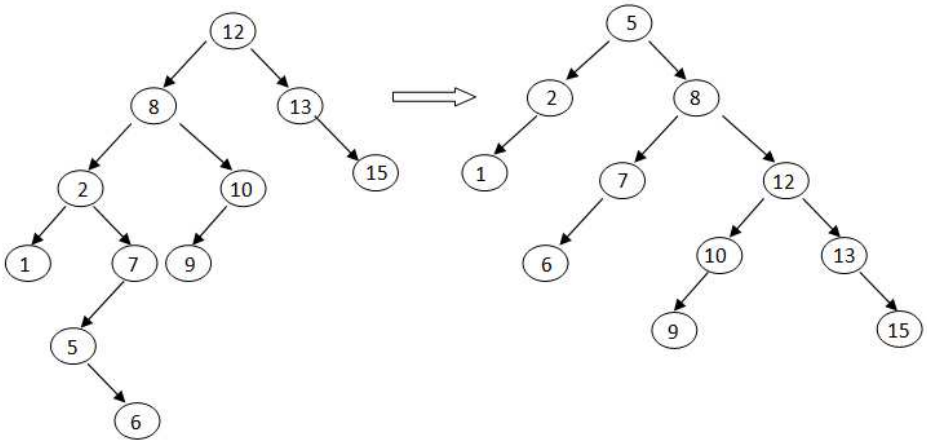
3. `moveToRoot (Branch 16 (Branch 2 (Branch 1 Empty Empty) (Branch 7 (Branch 3 Empty Empty) (Branch 10 Empty Empty))) (Branch 24 Empty Empty)) 7`

   `Branch 7 (Branch 2 (Branch 1 Empty Empty) (Branch 3 Empty Empty)) (Branch 16 (Branch 10 Empty Empty) (Branch 24 Empty Empty))`



4. `moveToRoot (Branch 12 (Branch 8 (Branch 2 (Branch 1 Empty Empty) (Branch 7 (Branch 5 Empty (Branch 6 Empty Empty)) Empty)) (Branch 10 (Branch 9 Empty Empty) Empty)) (Branch 13 Empty (Branch 15 Empty Empty))) 5`

   `Branch 5 (Branch 2 (Branch 1 Empty Empty) Empty) (Branch 8 (Branch 7 (Branch 6 Empty Empty) Empty) (Branch 12 (Branch 10 (Branch 9 Empty Empty) Empty) (Branch 13 Empty (Branch 15 Empty Empty))))`



4 Submission

Submission will be done via COW. You should upload a single Haskell file called "hw1.hs". Besides, you should test your codes in inek machines with hugs before submitting. Since black box method will be used in evaluation, be careful about the name of functions, data structures etc.

Late submission: At most 3 late days are allowed. After 3 days, you get 0.

5 Grading

This homework will be graded out of 100. It will make up 5% of your total grade.

6 Cheating Policy

Both parties involved in cheating get 0 from all of the 6 homeworks and will be reported to the university's disciplinary actions committee.