COL106: Major

QUESTION 12

(a) [10] A threaded BST (This is not about concurrency.) is a binary search tree that stores in each node with- out a right child, a reference to its in-order successor in place of the right null reference. Similarly, if the left reference would be null in a node, it stores a reference to the in-order predecessor instead. Two boolean flags at each node indicate for each reference if it is a tree references or a thread reference. For simplicity, you may assume separate references, *leftthread* and *rightthread*, are employed when left and right are null, respectively. Provide an algorithm to insert a key in a threaded BST. (You do not need to balance it.)
Solution:

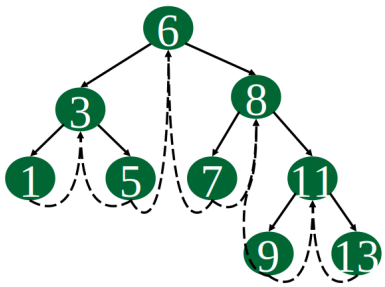

Fig. 1. A double threaded BST.
Source: http://web.eecs.umich.edu/ akamil/teaching/su02/080802.ppt

Let assume structure of the Node:

```
class Node {
   int data; // value
   int leftBit; // whether there is a value on left side. Defaut 0
   int rightBit; // whether there is a value on right side. Defaut 0
   Node left; // pointer to node on left
   Node right; // pointer to node on right
}
```

(1) To insert a node our first task is to find the place to insert the node.
(2) First check if tree is empty, means tree has just dummy node then then insert the new node into left subtree of the dummy node.
(3) If tree is not empty then find the place to insert the node, just like in normal BST.
(4) If new node is smaller than or equal to current node then check if *leftBit =0*, if yes then we have found the place to insert the node, it will be in the left of the subtree and if *leftBit=1* then go left.
(5) If new node is greater than current node then check if *rightBit =0*, if yes then we have found the place to insert the node, it will be in the right of the sub-tree and if *rightBit=1* then go right.
(6) Repeat step 4 and 5 till the place to be inserted is not found.

(7) Once decided where the node will be inserted, next task would be to insert the node. first we will see how the node will be inserted as left child.

(8) Insert as left child (new node *n*)

```
n.left = current.left;
current.left = n;
n.leftBit = current.leftBit;
current.leftBit = 1;
n.right = current; // update the thread
```

(9) Insert as right child

```
n.right = current.right;
current.right = n;
n.rightBit = current.rightBit;
current.rightBit = 1;
n.left = current; // update the thread
```

(b) [4] Is there any advantage of using threaded AVL tree in the problem in Qn 11? Explain.
Solution: Based on the data structure used previously, it will help in the find operation and insert operations. Run-time analysis will depend on the data structure used.