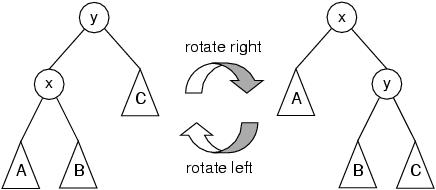
**CME227 DATA STRUCTURES**

**Final, January, 06, 2014**

**Number/Name/Division:....................................................................... Duration: 75 minutes**

1-) (25 pts.) Let q be a non-empty queue and let s be an empty stack. Using only the stack and queue, write a function ***reverse(queue q, stack s)*** to reverse the order of elements in q.

2-) (25 pts.) Write a function that returns the degree of a given vertex **v** in an undirected graph which is represented by an adjacency list.

3-) (20 pts.) We suppose that an AVL tree node has the following data structure.

struct node{  
 int key;  
 struct node \*left;  
 struct node \*right;  
 int height;  
};

Complete the following function that performs right rotation. Note that height member of some nodes should be updated.

struct node \*rightRotate(struct node \*y){

…………………

return ….

}

4-) (25 pts) Draw the last Binary Search Tree after inserting the following key values in the given order into an empty BST. Then perform inorder, preorder and postorder traversals.

100, 56, 92, 160, 92, 90, 192, 28, 20, 136, 40, 30, 68, 144

5-) (25 pts.) Circle T or F for each of the following statements to indicate whether the statement is **true** or **false**, respectively. If the statement is wrong, explain why.

* **T F** The height of any binary search tree with n nodes is O(log n)
* **T F** The heights of any two siblings in a binary heap differ by at most 1.
* **T F** A subtree of the root of an AVL tree is always itself an AVL tree
* **T F** An AVL tree is always the same after deletion and then insertion the same node
* **T F** A tree is a graph that may have cycle
* **T F** Adjacency list representation of graphs is more efficient than adjacency matrix representation for sparse graphs.
* **T F** The array [6 ; 8 ; 15 ; 11 ; 9 ; 12] is a binary min heap. Indices start with 1.
* **T F** Recursive function calls use queue
* **T F** A binary heap is also an AVL tree
* **T F** The length of a path is the number of edges on that path