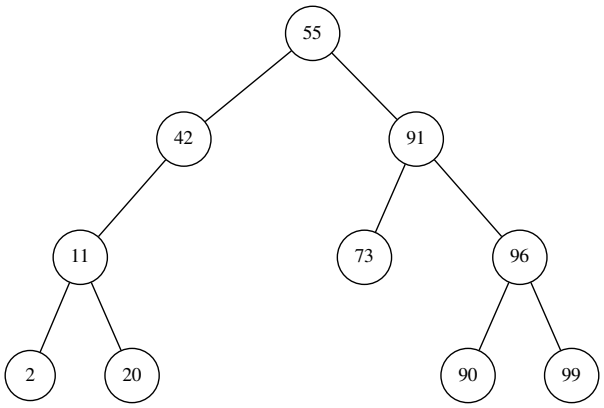
CSC 212 Midterm 2 - Fall 2013

College of Computer and Information Sciences, King Saud University
Exam Duration: 2 Hours

17/12/2013

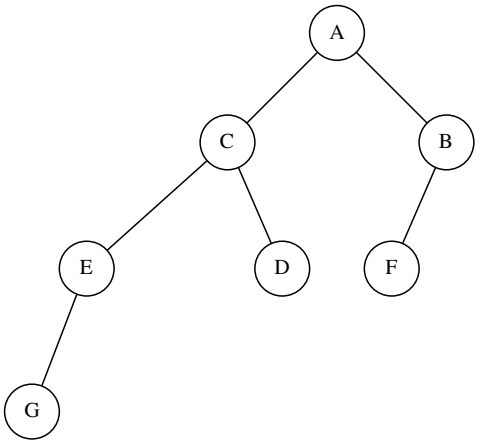
Question 1 [25 points]

1. Write the order of traversing the binary tree below using **inorder** and **postorder** traversals:



2. Write a main method that uses the class BT to create the binary tree represented below. The values must be inserted in the following order: A, B, C, D, E, F, G.

```
public static void main(String[] args){
        BT<String> tree = new BT<String>();
        ...
}
```



. . . . . .

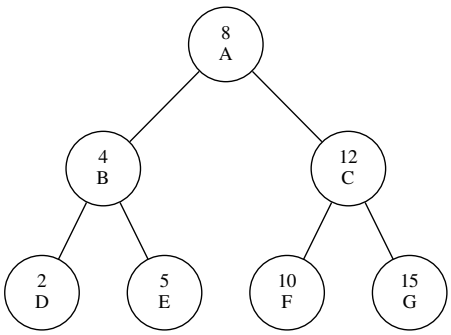
Question 2 [25 points]

1. Write an efficient method *inRange*, member of the class *BST*, that takes as in put a key k and returns true if the binary search tree contains at least two keys k_1 and k_2 such that $k_1 \leq k \leq k_2$, false otherwise. Try to minimize the number of visited nodes. The method signature is: *public boolean inRange(int k)*.

   For **example**, for the tree below, *inRange(3)* returns true, *inRange(2)* returns true, *inRange(1)* returns false and *inRange(20)* returns false.

2. Write the **recursive** method *rangeFind*, member of the class *BST*, that takes as input two keys k_1 and k_2 and returns a list that contains all the data in the tree with keys k satisfying: $k_1 \leq k \leq k_2$ (assume that $k_1 \leq k_2$). The order of the data in the list must be the same as that of the keys. The method signature is *public List< T >rangeFind (int k_1, int k_2)*. The method must call the private method *recRangeFind*.

   For **example**, the call *rangeFind(4,10)* on the tree below returns the list: $B \rightarrow E \rightarrow A \rightarrow F$. The call *rangeFind(9,20)* returns: $F \rightarrow C \rightarrow G$.



. . . . . .

Question 3 [25 points]

1. A heap is stored in the array below. Answer the following:

Position	0	1	2	3	4	5	6	7	8	9	10
Key	-	15	23	18	26	33	21	70	67	28	34

   (a) What is the position of the left child of 26 in the array?
   (b) What is the position of the right child of 26 in the array?
   (c) What is the position of the parent of 28 in array?
   (d) Is it a min-heap or a max-heap?

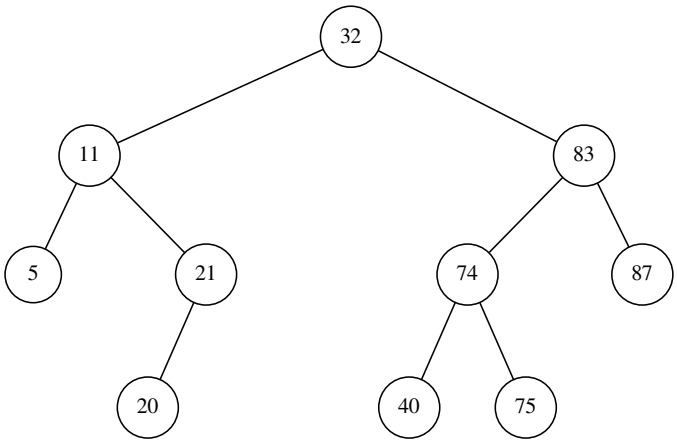
2. Insert the following keys in a **max-heap**: 7, 14, 18, 24, 47, 35, 11 and 57. The heap is initially empty. Insert the keys from left to right. Draw the heap **after every insert**.

3. **Delete two** keys from the heap. Draw the heap **after every delete** operation.

. . . . . .

Question 4 [25 points]

Perform the following operations on the AVL tree below: (1) **Insert 24**. (2) **Insert 78**. (3) **Insert 35**. (4) **Remove 32**. (5) **Remove 87**. Each operation is **independent** of the others and must be performed on the original tree. Make sure to mention the **rotation** performed (none, single, double).



. . . . . .

A ADT Binary Tree Specification

- boolean **find** (Relative rel): **Requires**: BT is not empty. **Results**: the current node of BT is determined by Relative and the current node prior to the operation as follows (always return true unless indicated so): (1) rel = Root: current = root (2) rel = Parent: if the current node has a parent then parent is the current node; otherwise returns false (3) rel = LeftChild: if the current node has a leftchild then it will be the current node; otherwise returns false (4) rel = RightChild: same as above but for rightchild.

- boolean **insert**(Relative rel, Type val): **Requires**: either (1) BT is empty and rel = Root; or (2) BT not empty and rel= Root . **Results**: as follows: (1) rel = Root: create a root node with data = val. (2) rel = Parent: nonsense case. (3) rel = LeftChild: if current node does not have a leftchild then make one with data = val. (4) rel = RightChild: same as above but for rightchild. In all the above cases if the insertion was successful then it will be designated as current node and returns true, otherwise current remains unchanged and returns false.

B ADT List Specification

- void **insert** (Type e): **Requires**: list L is not full. **Results**: a new node containing element e is created and inserted after the current element in the list. The new element e is made the current element. If the list is empty e is also made the head element.