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. Order of traversal:
   - Inorder: 2, 11, 20, 42, 55, 73, 91, 90, 96, 99.
   - Postorder: 2, 20, 11, 42, 73, 90, 99, 96, 91, 55.

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> t = new BT<String>();
        t.insert("A", Relative.Root);
        t.insert("B", Relative.RightChild);
        t.find(Relative.Parent);
        t.insert("C", Relative.LefttChild);
        t.insert("D", Relative.RightChild);
        t.find(Relative.Parent);
        t.insert("E", Relative.LefttChild);
        t.find(Relative.Root);
        t.find(Relative.RightChild);
        t.insert("F", Relative.LefttChild);
        t.find(Relative.Root);
        t.find(Relative.LeftChild);
        t.find(Relative.LeftChild);
        t.insert("G", Relative.LefttChild);
}
```

. . . .

Question 2 [25 points]

1.
```
// A simple solution
public boolean inRange(int k){

        if(root == null) // Empty tree
                return false;

        BTNode<T> p= root;
        // Find min
        while(p.left != null){
                p= p.left;
        }

        int min= p.key;

        p= root;
        // Find max
        while(p.right != null){
                p= p.right;
        }

        int max= p.key;

        if((min <= k) && (k <= max))
                return true;
        else
                return false;
}

// A better solution
public boolean inRange(int k){

        if(root == null) // Empty tree
                return false;

        BTNode<T> p= root;

        // Looking for k1
        while((p.key > k) && (p.left != null)){
                p= p.left;
        }
        if(p.key>k)
                return false;

        // Looking for k2
        while((p.key < k) && (p.right != null)){
                p= p.right;
        }

        if(p.key<k)
                return false;

        return true;
}
```

2.

```
public List<T> rangeFind(int k1, int k2){

        List<T> l= new List<T>();
        recRangeFind(root, l, k1, k2);
        return l;
}

private void recRangeFind(BTNode<T> t, List<T> l, int k1, int k2){

        if(t == null)
                return;
        if(k1 < t.key) // Check the left subtree
                recRangeFind(t.left, l, k1, k2);

        if ((k1 <= t.key) && (t.key <= k2)) // Insert t.data
                l.insert(t.data);

        if(t.key < k2) // Check the right subtree
                recRangeFind(t.right, l, k1, k2);
}
```

....

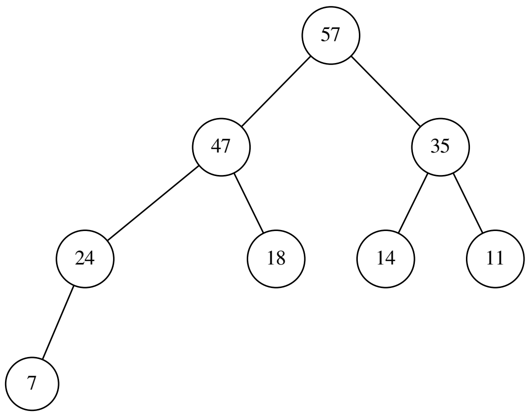
Question 3 [25 points]

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

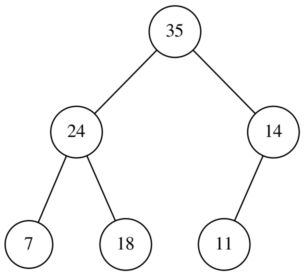
Position	0	1	2	3	4	5	6	7	8	9	10
Key	-	15	23	18	26	33	21	70	67	28	34

   (a) 8.
   (b) 9.
   (c) 4.
   (d) Min-heap.

2. Heap after all inserts:



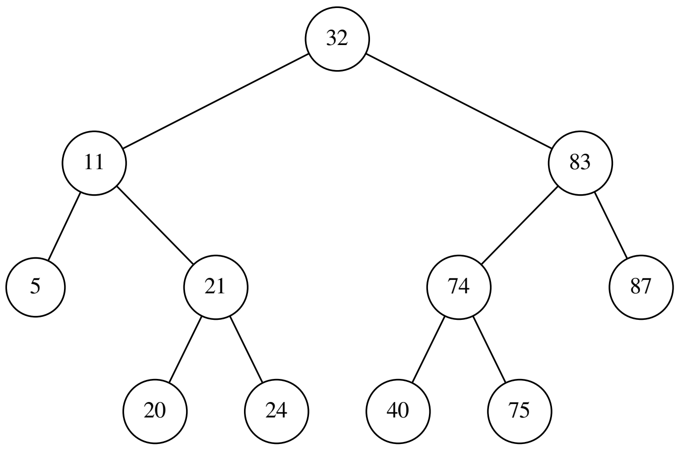
3. Heap after all deletes:



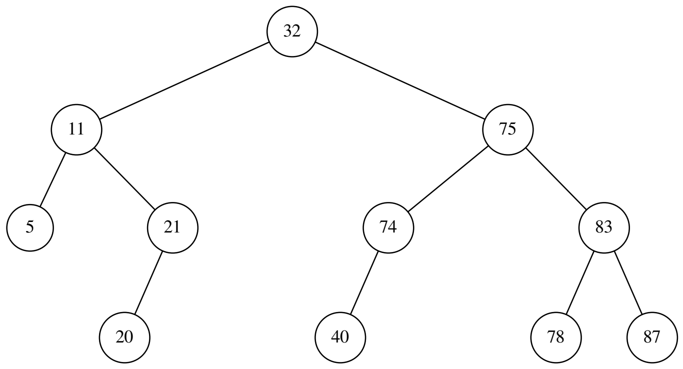
. . . .

Question 4 [25 points]

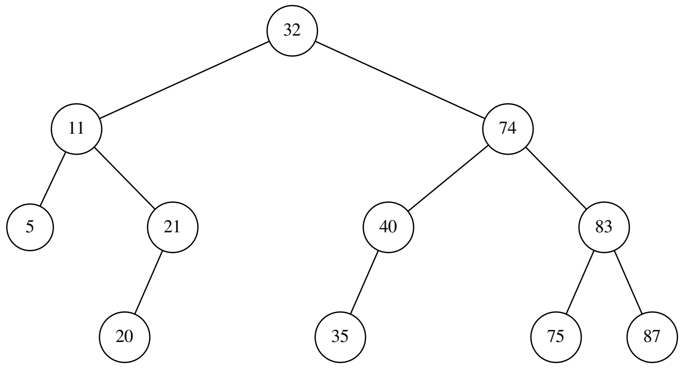
- **Insert 24** (none).



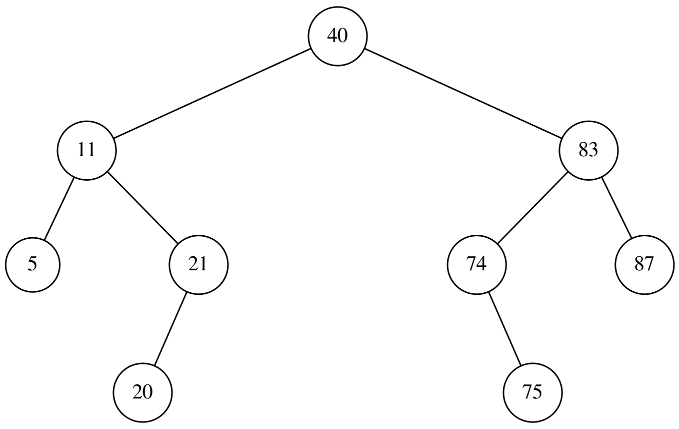
- **Insert 78** (double).



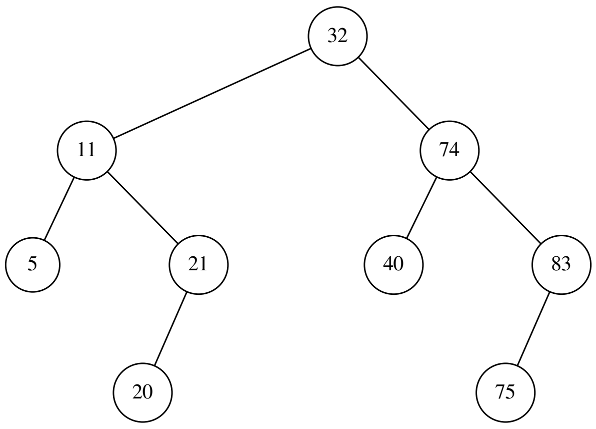
- **Insert 35** (single).



- **Remove 32** (none).



- **Remove 87** (single).



. . . .