King Saud University
College of Computer and Information Sciences
Computer Science Department

CSC 212 **Second Semester 1439-1440**

Tutorial #9

Problem 1

Write the recursive method ***sumKeys*** member of the class BST that returns the sum of all the keys in the tree.

Method: public int sumKeys()

```
public int sumKeys() {
     return sumKeysRec(root);
}

private int sumKeysRec(BSTNode<T> pointer) {
     if (pointer == null)
          return 0;
     return pointer.key + sumKeysRec(pointer.left) +
     sumKeysRec(pointer.right);
}
```

Problem 2

Write the method ***getRange*** member of the class BST that returns the range of the binary search tree. The range is defined as the difference between the maximum key and the minimum key. Assume that the tree is not empty.

Method: public int getRange()

```
public int getRange() {
     BSTNode<T> pointer = root;
     int minKey, maxKey;
     while (pointer.left != null)
          pointer = pointer.left;
     minKey = pointer.key;
     pointer = root;
     while (pointer.right != null)
          pointer = pointer.right;
     maxKey = pointer.key;
     return maxKey - minKey;
}
```

Problem 3

Write an efficient method ***isInRange***, member of the class BST, that takes as input a key ***key*** and returns true if the binary search tree contains at least two keys ***lowerKey*** and ***upperKey*** such that ***lowerKey ≤ key ≤ upperKey***, false otherwise. Try to minimize the number of visited nodes.

Method: public boolean isInRange(int key)

```
    // Solution #1
    public boolean isInRange(int key) {
        if (root == null)
            return false;
        BSTNode<T> pointer = root;
        int lowerKey, upperKey;
        while (pointer.left != null)
            pointer = pointer.left;
        lowerKey = pointer.key;
        pointer = root;
        while (pointer.right != null)
            pointer = pointer.right;
        upperKey = pointer.key;
        return lowerKey <= key && key <= upperKey;
    }
    // Solution #2 (More efficient)
    public boolean isInRange(int key) {
        if (root == null)
            return false;
        BSTNode<T> pointer = root;
        Boolean inRange = false;
        while (pointer.left != null) {
            if (pointer.key <= key) {
                inRange = true;
                break;
            }
            pointer = pointer.left;
        }
        if (!inRange)
            return false;
        pointer = root;
        inRange = false;
        while (pointer.right != null) {
            if (key <= pointer.key) {
                inRange =true;
                break;
            }
            pointer = pointer.right;
        }
        if (!inRange)
            return false;
        return true;
    }
```

Problem 4

Write the member method ***countNodesIn*** member of the class BST that returns the number of nodes in the subtree rooted at the node with key ***key***. Assume that ***key*** exists. You are not allowed to call any of the ***BST*** methods.

Method: public int countNodesIn(int key)

```
public int countNodesIn(int key) {
	BSTNode<T> pointer = root;
	while (pointer != null) {
		if (pointer.key == key)
			break;
		else if (key < pointer.key)
			pointer = pointer.left;
		else
			pointer = pointer.right;
	}
	return countNodesInRec(pointer);
}

private int countNodesInRec(BSTNode<T> pointer) {
	if (pointer == null)
		return 0;
	return countNodesInRec(pointer.left) +
	countNodesInRec(pointer.right) + 1;
}
```

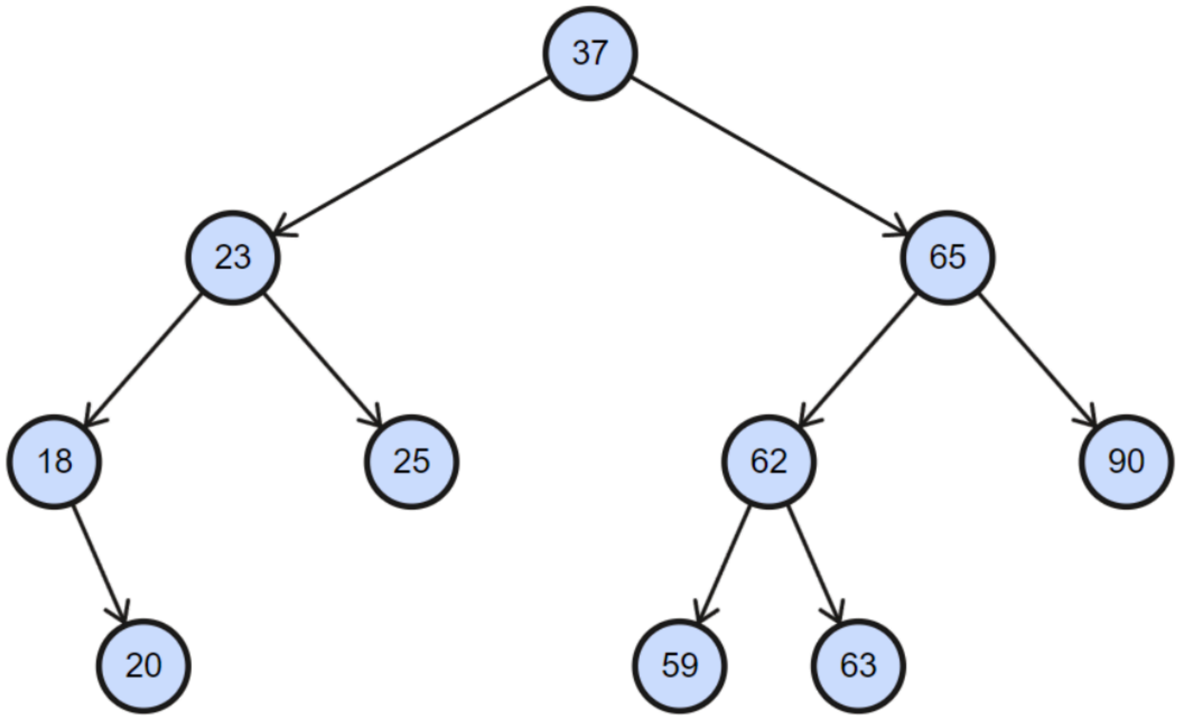
Problem 5

A) Insert the following keys into an empty binary search tree: 37, 23, 18, 65, 25, 62, 20, 59, 63, 90, 18.

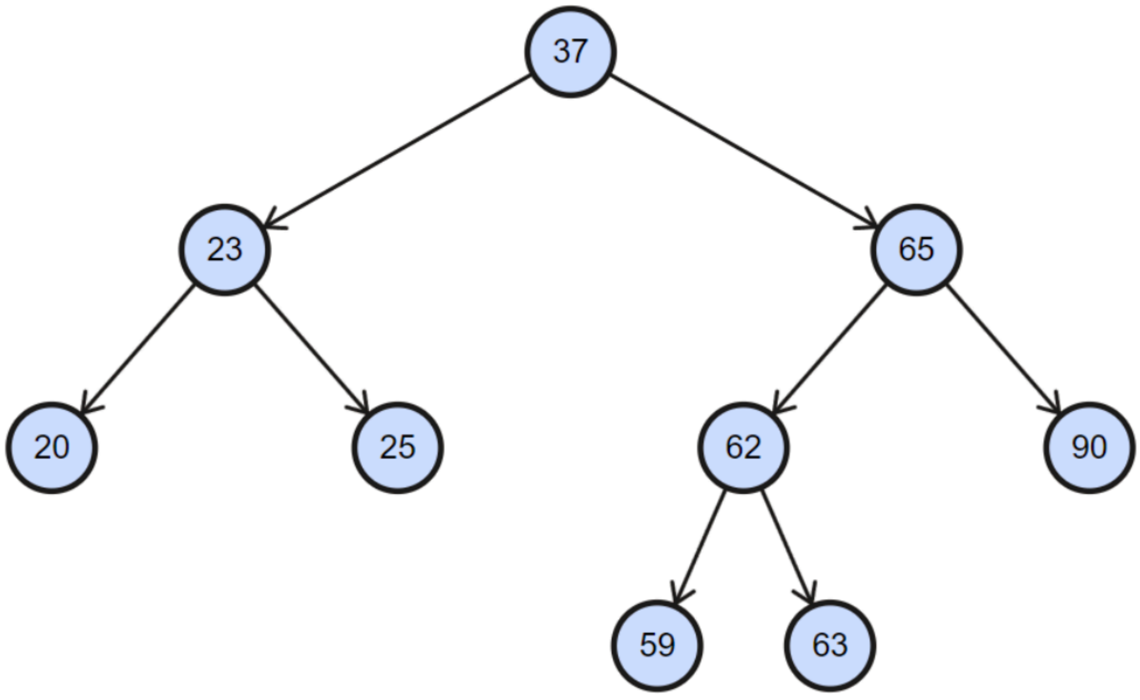
B) Remove the following keys from the final tree in part A: 18, 90, 37.

C) If we wish to print the keys in increasing order, which traversal method should we use?

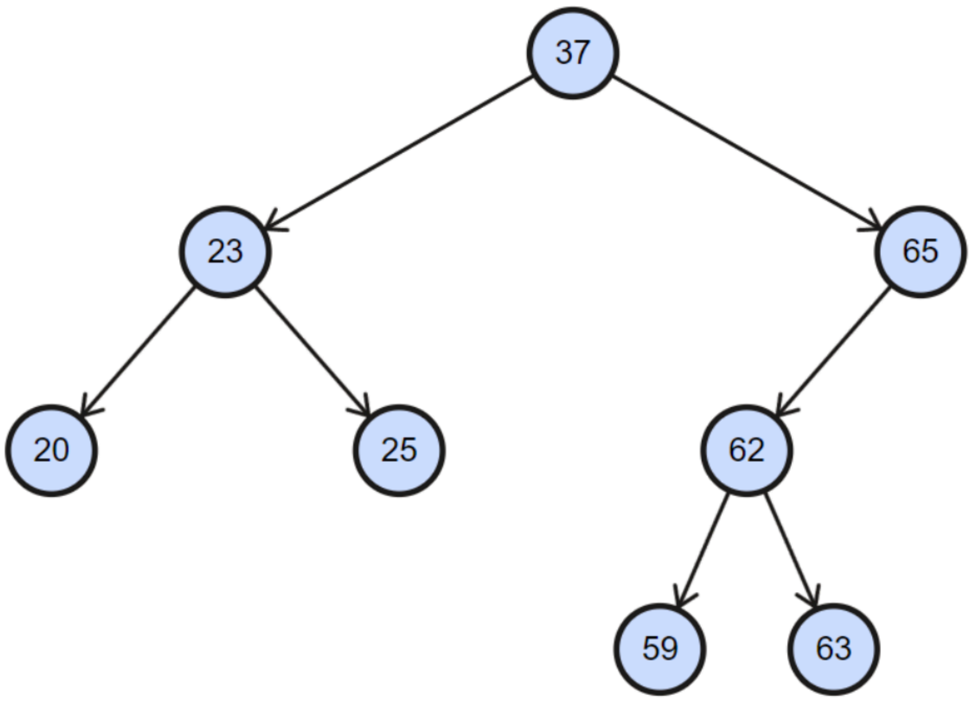
Solution:
A)



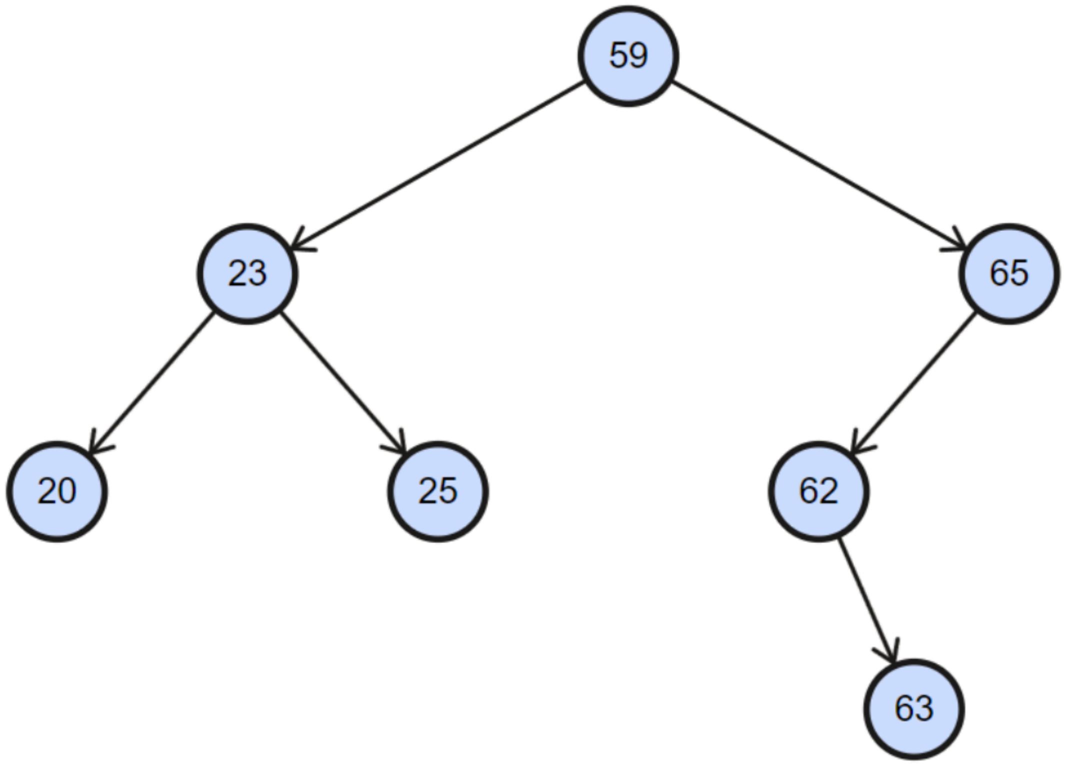
B) Removing 18 (case 2: having one child)



Removing 90 (case 1: having no children)



Removing 37 (case 3: having two children; left-most node in the right sub-tree)



C) Inorder (left-root-right)