# ICS-202 Lab-06 Report

Mohammed Busaleh - 202158210

## Exercise 1:

### Code:

```java
//HERE STARTS MY LAB WORK

public int count() {
    if(this.isEmpty())
        return 0;

    return countHelper(root);
}

private int countHelper(BSTNode<T> node){
    if(node == null)
        return 0;
    else if(node.left == null)
        return 1 + countHelper(node.right);
    else if(node.right == null)
        return 1+ countHelper(node.left);
    else
        return 1 + countHelper(node.left) + countHelper(node.right);

}
```

## Exercise 2:

### Code:

```java
public boolean isLeaf(T value){
    if(isEmpty())
        throw new RuntimeException("Empty BST");
    else if(!isInTree(value))
        return false;

    BSTNode<T> p = root;
    boolean found = false;
    while (!found)
        if (value.equals(p.el))
            found = true;
        else if (value.compareTo(p.el) < 0)
            p = p.left;
        else p = p.right;

    return p.left == null && p.right == null;
}
```

## Exercise 3:

### Code:

```java
public int countLeaves(){
    if(isEmpty())
        throw new RuntimeException("Empty BST");

    return countLeavesHelper(root);
}

private int countLeavesHelper(BSTNode<T> node){
    if(isLeaf(node.el))
        return 1;
    else if(node.left != null && node.right != null)
        return countLeavesHelper(node.left) + countLeavesHelper(node.right);
    else if(node.left == null)
        return countLeavesHelper(node.right);
    else
        return countLeavesHelper(node.left);
}
```

## Exercise 4:

### Code:

```java
public int height(){
    if(isEmpty())
        throw new RuntimeException("Empty BST");

    return heightHelper(root);
}

private int heightHelper(BSTNode<T> node){
    if(node.left == null && node.right == null)
        return 0;
    else if(node.left == null)
        return 1 + heightHelper(node.right);
    else if(node.right == null)
        return 1 + heightHelper(node.left);
    else
        return 1 + Math.max(heightHelper(node.left), heightHelper(node.right));
}
```

## Exercise 5:

### Code:

```
public static void main(String[] args) {
    BST<Integer> myBST = new BST<>();

    for(int i = 0; i < 10; i++)
        myBST.insert((int)(1+Math.random()*100));

    System.out.println("The number of nodes is: " + myBST.count() +
                       "\n'4' is leaf? " + myBST.isLeaf(4) +
                       "\n'7' is leaf? " + myBST.isLeaf(7) +
                       "\nNumber of leaves is: " + myBST.countLeaves() +
                       "\n\nThe various traversals are: ");
    System.out.print("Preorder: ");
    myBST.preorder();
    System.out.print("\nInorder: ");
    myBST.inorder();
    System.out.print("\nPostorder: ");
    myBST.postorder();
    System.out.print("\nBreadth-First: ");
    myBST.breadthFirst();

    System.out.println("\n\n\n");
}
```

### Output:

```
The number of nodes is: 10
'4' is leaf? false
'7' is leaf? true
Number of leaves is: 4

The various traversals are:
Preorder: 99 23 10 7 61 44 38 60 93 71
Inorder: 7 10 23 38 44 60 61 71 93 99
Postorder: 7 10 38 60 44 71 93 61 23 99
Breadth-First: 99 23 10 61 7 44 93 38 60 71
```