UNIVERSITI TEKNOLOGI MARA

ASSIGNMENT #2

COURSE: CSC508

GROUP: CDCS2304B

LECTURER: DR. NUR FARRALIZA BINTI MANSOR

NAME	STUDENT ID
AMIR FIRDAUS BIN MOHAMAD NAFIAH	2024542677
MUHAMMAD ADAM HAQIMI BIN AHMAD NAFIAH	2024916833
DEAN ARDLEY BIN REZALI	2024793015

QUESTION

1. Define your own Binary Search Tree (BST) data structure using Linked List. You can name your BST as **MyBST**.

```
public class TreeNode {
    TreeNode left;
    TreeNode right;
    int data;

    //constructor
    public TreeNode(int data)
    {
        this.data = data;
        left = right = null;
    }

    public void insert(int value)
    {
        if (value < data) {
            if (left == null) {
                left = new TreeNode(value);
            } else {
                left.insert(value);
            }
        } else if (value > data) {
            if (right == null) {
                right = new TreeNode(value);
            } else {
                right.insert(value);
            }
        }
    }
}
```

2. Define methods in the BST to perform the following operations :

    a. Determine whether the BST is empty

```
//Checkk if myBST is empty
public boolean isEmpty()
{
    return root == null;
}
```

    b. Insert an item in the BST.

```
//insert a new node into the tree
public void insert(int value)
{
    if(root == null){
        root = new TreeNode (value);
    }
    else{
        root.insert(value);
    }
}
```

c. Traverse the BST (preorder, inorder, postorder)

```
//preorder
public void preorder()
{
    preorderPTE(root);
}

private void preorderPTE(TreeNode node)
{
    if(node != null){
        System.out.println(node.data + " ");
        preorderPTE(node.left);
        preorderPTE(node.right);
    }
}
```

```
//inorder
public void inorder()
{
    inorderPTE(root);
}

private void inorderPTE(TreeNode node)
{
    if(node != null){
        inorderPTE(node.left);
        System.out.println(node.data + " ");
        inorderPTE(node.right);
    }
}
```

```
//postorder
public void postorder()
{
    postorderPTE(root);
}

private void postorderPTE(TreeNode node)
{
    if(node != null){
        postorderPTE(node.left);
        postorderPTE(node.right);
        System.out.println(node.data + " ");
    }
}
```

d. Calculate the height of the BST

```java
//calculate the height of myBST
public int height()
{
    return heightPTE(root);
}

private int heightPTE(TreeNode node)
{
    if(node == null)
    {
        return 0;
    }
    else
    {
        return 1 + Math.max(heightPTE(node.left), heightPTE(node.right));
    }
}
```

e. Find the level of a selected node in the BST

```java
//find the level of a selected node
public int getLevel(int value)
{
    return getLevelPTE(root, value, 1);
}

private int getLevelPTE(TreeNode node, int value, int level)
{
    if(node == null)
    {
        return 0;
    }
    if(node.data == value){
        return level;
    }

    int downLevel = getLevelPTE(node.left, value, level + 1);

    if(downLevel != 0){
        return downLevel;
    }

    return getLevelPTE(node.right, value, level + 1);
}
```

f. Calculate the number of nodes in the BST

```java
//calculate the number of nodes in myBST
public int countNode()
{
    return countNodePTE(root);
}

private int countNodePTE(TreeNode node)
{
    if(node == null)
    {
        return 0;
    }

    return 1 + countNodePTE(node.left) + countNodePTE(node.right);
}
```

g. Calculate the number of leaves in the BST

```java
//calculate the number of leaves in myBST
public int countLeave()
{
    return countLeavePTE(root);
}

private int countLeavePTE(TreeNode node)
{
    if(node == null)
    {
        return 0;
    }

    if (node.left == null && node.right == null)
    {
        return 1;
    }

    return countLeavePTE(node.left) + countLeavePTE(node.right);
}
```

h. Calculate the minimum/maximum value in the BST

```java
//find the minimum value in myBST
 public int findMin() {
    if (root == null)
    {
        System.out.println("BST is empty!");
    }
    TreeNode current = root;
    while (current.left != null) {
        current = current.left;
    }
    return current.data;
}

 //find teh maximum value in myBst
 public int findMax() {
    if (root == null)
    {
        System.out.println("BST is empty!");
    }
    TreeNode current = root;
    while (current.right != null) {
        current = current.right;
    }
    return current.data;
}
```

i. Calculate the total/average value in the BST

```java
//calc total value of nodes in myBST
 public int calculateTotal() {
    return calcNodePTE(root);
}


private int calcNodePTE(TreeNode node)
{
    if(node == null)
    {
        return 0;
    }

    return node.data + calcNodePTE(node.left) + calcNodePTE(node.right);
}

//calc avg value of nodes in myBST
public double calculateAverage()
{
    int total = calculateTotal();
    int count = countNode();

    return total / count;
}
```

3. Write the application class which contain main() to declare object **bst** from class **MyBST**.

```java
public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);
    myBST bst = new myBST();
```

4. Ask the user to enter at least 20 numbers into the BST.

```java
System.out.println("Enter 20 numbers to insert into the BST:");
for (int i = 0; i < 20; i++) {
    int num = sc.nextInt();
    bst.insert(num);
}
```

5. Call the following methods (c – i) from main() and display the results accordingly.

```
System.out.println("\nInorder traversal:");
bst.inorder();

System.out.println("\nPreorder traversal:");
bst.preorder();

System.out.println("\nPostorder traversal:");
bst.postorder();

System.out.println("\nHeight of BST: " + bst.height());
System.out.println("Number of nodes: " + bst.countNode());
System.out.println("Number of leaves: " + bst.countLeave());
System.out.println("Minimum value: " + bst.findMin());
System.out.println("Maximum value: " + bst.findMax());
System.out.println("Total value: " + bst.calculateTotal());
System.out.println("Average value: " + bst.calculateAverage());
```

Output:

```
Enter 20 numbers to insert into the BST:
30
71
52
29
79
68
98
73
38
14
23
64
51
77
90
33
43
10
81
22
```

```
Inorder traversal:
10
14
22
23
29
30
33
38
43
51
52
64
68
71
73
77
79
81
90
98
```

```
Preorder traversal:
30
29
14
10
23
22
71
52
38
33
51
43
68
64
79
73
77
98
90
81
```

```
Postorder traversal:
10
22
23
14
29
33
43
51
38
64
68
52
77
73
81
90
98
79
71
30
```

```
Height of BST: 6
Number of nodes: 20
Number of leaves: 7
Minimum value: 10
Maximum value: 98
Total value: 1046
Average value: 52.0
```

```
Enter a number to find its level:
30
Level of 30: 1
```