Lab #10: Tree

The main aim of the lab is to get familiar with Tree structure, especially Binary Tree Search.

Deadline: 23h59, 04/12/2023

For give BNode, BST classes are as follows:

```java
public class BNode<E extends Comparable<E>> {
     private E data;
     private BNode<E> left;
     private BNode<E> right;

     public BNode(E data) {
          this.data = data;
          this.myLeft = null;
          this.myRight = null;
     }

     public BNode(E data, BNode<E> left, BNode<E> right) {
          this.data = data;
          this.myLeft = left;
          this.myRight = right;
     }

}
```

Binary Search Tree class:

```java
public class BST<E extends Comparable<E>> {
     private BNode<E> root;

     public BST() {
          this.root = null;

     }

}
```

==

Task 1: Basic Operations with BST

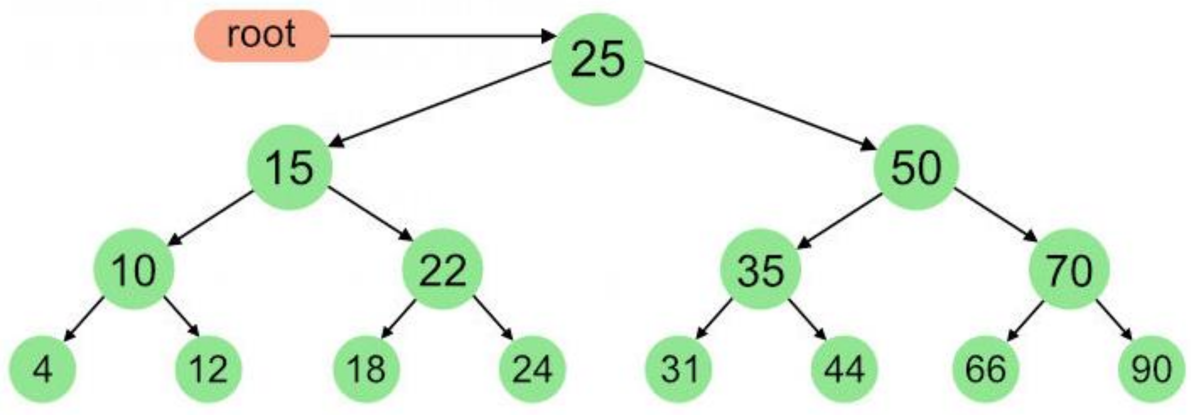
==

Implement the following methods in the **BST** class:

```
// Add element e into BST
     public void add(E e) {
          // TODO
     }

     // Add a collection of elements col into BST
     public void add(Collection<E> col) {
          // TODO
     }

     // compute the depth of a node in BST
     public int depth(E node) {
          // TODO
          return -1;
     }

     // compute the height of BST
     public int height() {
          // TODO
          return -1;
     }

     // Compute total nodes in BST
     public int size() {
          // TODO
          return -1;
     }

     // Check whether element e is in BST
     public boolean contains(E e) {
          // TODO
          return false;
     }

     // Find the minimum element in BST
     public E findMin() {
          // TODO
          return null;
     }

     // Find the maximum element in BST
     public E findMax() {
          // TODO
          return null;
     }
```

```
    // Remove element e from BST
    public boolean remove(E e) {
        // TODO
        return false;
    }
// get the descendants of a node
    public List<E> descendants(E data) {
        // TODO
        return null;
    }

    // get the ancestors of a node
    public List<E> ancestors(E data) {
        // TODO
        return null;
    }
```

For testing, use the following BST:



==

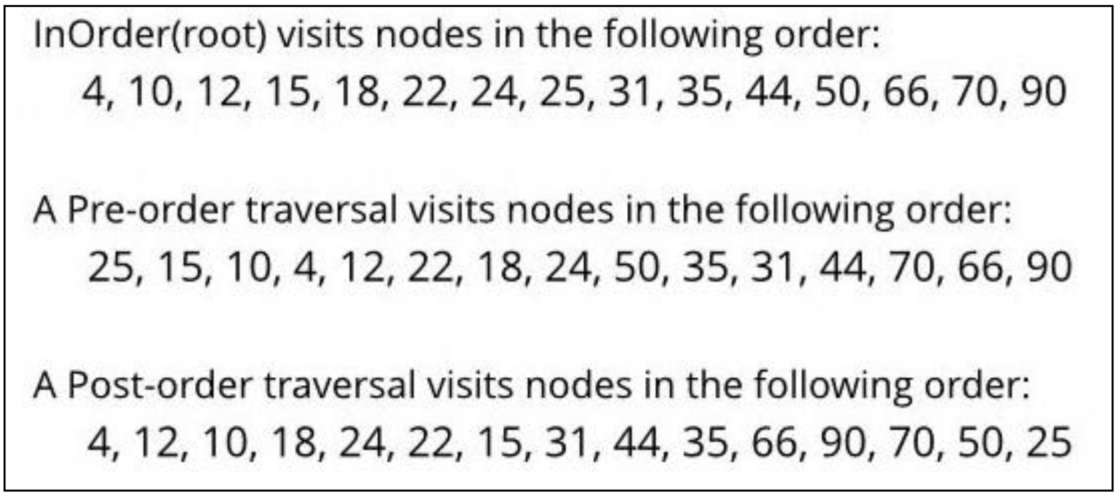
Task 2: Tree Traversal Algorithms

==

Implement the following methods in the **BST** class:

```
// display BST using inorder approach
    public void inorder() {
        // TODO
    }

    // display BST using preorder approach
    public void preorder() {
        // TODO
    }

    // display BST using postorder approach
    public void postorder() {
        // TODO
    }
```

Using the above BST, traversal results are as follows:



InOrder(root) visits nodes in the following order:
4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90

A Pre-order traversal visits nodes in the following order:
25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90

A Post-order traversal visits nodes in the following order:
4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25