Name: Sujitkumar Sanjay Patil
Roll No.:222053
Prn. No.: 22010360
Class: SY B2

Assignment 1

Aim: Implement binary search tree and perform following operations: a. Insert, Delete, Display

Code:

```
import java.util.*;

class Main {

  class Node {
    int key;
    Node left, right;

    public Node(int item) {
      key = item;
      left = right = null;
    }
  }

  Node root;

  Main() {
    root = null;
  }

  void insert(int key) {
    root = insertKey(root, key);
  }

  // Insert key in the tree
  Node insertKey(Node root, int key) {
    // Return a new node if the tree is empty
    if (root == null) {
      root = new Node(key);
      return root;
    }

    // Traverse to the right place and insert the node
    if (key < root.key)
      root.left = insertKey(root.left, key);
```

```
    else if (key > root.key)
      root.right = insertKey(root.right, key);

    return root;
  }

  void inorder() {
    inorderRec(root);
  }

  // Inorder Traversal
  void inorderRec(Node root) {
    if (root != null) {
      inorderRec(root.left);
      System.out.print(root.key + " -> ");
      inorderRec(root.right);
    }
  }

  void deleteKey(int key) {
    root = deleteRec(root, key);
  }

  Node deleteRec(Node root, int key) {
    // Return if the tree is empty
    if (root == null)
      return root;

    // Find the node to be deleted
    if (key < root.key)
      root.left = deleteRec(root.left, key);
    else if (key > root.key)
      root.right = deleteRec(root.right, key);
    else {
      // If the node is with only one child or no child
      if (root.left == null)
        return root.right;
      else if (root.right == null)
        return root.left;

      // If the node has two children
      // Place the inorder successor in position of the node to be deleted
      root.key = minValue(root.right);

      // Delete the inorder successor
      root.right = deleteRec(root.right, root.key);
    }
```

```
    return root;
  }

  // Find the inorder successor
  int minValue(Node root) {
    int minv = root.key;
    while (root.left != null) {
      minv = root.left.key;
      root = root.left;
    }
    return minv;
  }

  // Driver Program to test above functions
  public static void main(String[] args) {
    Scanner sc= new Scanner(System.in);

    Main tree = new Main();

    while(true){
    System.out.println("\t\tOption Menu");
    System.out.println("1.Insert\n2.Delete\n3.Display\n4.Quit\n");
    System.out.print("Enter your choice: ");
    int opt = sc.nextInt();
    if(opt==1){
        System.out.print("Enter number to insert in BST: ");
        int a = sc.nextInt();
        tree.insert(a);
        }
    else if(opt==2){
        System.out.print("Enter number to delete from BST: ");
        int b = sc.nextInt();
        tree.deleteKey(b);
    }
    else if(opt==3){
        System.out.print("Inorder traversal: ");
        tree.inorder();
        System.out.println("\n");
    }
    else if(opt==4){
        break;
    }
    else{
        System.out.println("Invalid Input");
    }
    }
  }
}
```

Output:

```
                Option Menu
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice: 1
Enter number to insert in BST: 45
                Option Menu
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice: 1
Enter number to insert in BST: 65
                Option Menu
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice: 3
Inorder traversal: 45 -> 65 ->
```

```
                Option Menu
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice: 2
Enter number to delete from BST: 45
                Option Menu
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice: 3
Inorder traversal: 65 ->

                Option Menu
1.Insert
2.Delete
3.Display
4.Quit

Enter your choice: 4


...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion: Thus we successfully created BST and perform the search , insert, delete and display operations on BST.