PROGRAM NO: 9 DATE: 19-10-2023

**AIM**

To implement Binary Search Tree with Insertion, Inorder, Postorder and Preorder Traversal.

**ALGORITHM**

1. START
2. FUNC CreateNode(value):
3. DECLARE node with malloc()
4. SET node.value = value
5. SET node.left = NULL
6. SET node.right = NULL
7. RETURN node, END FUNC
8. FUNC INSERT(value):
9. DECLARE newNode = this.CreateNode(value)
10. DECLARE root = this.Root
11. DECLARE prevRoot = NULL
12. DECLARE WHILE root != NULL:
13. SET prevRoot = root
14. IF newNode.value <= root.value:
15. SET root = root.left
16. ELSE: SET root = root.right
17. END WHILE
18. IF newNode.value <= prevRoot.value: SET prevRoot.left = newNode
19. ELSE: SET prevRoot.right = newNode
20. END FUNC
21. FUNC In\_Order\_Traversal(root)
22. IF root == NULL: RETURN
23. END IF
24. INVOKE In\_Order\_Traversal(root.left)
25. Print root.value
26. INVOKE In\_Order\_Traversal(root.right)
27. END FUNC
28. FUNC Pre\_Order\_Traversal(root)
29. IF root == NULL: RETURN
30. END IF
31. Print root.value
32. INVOKE Pre\_Order\_Traversal(root.left)
33. INVOKE Pre\_Order\_Traversal(root.right)
34. END FUNC
35. FUNC Post\_Order\_Traversal(root)
36. IF root == NULL: RETURN
37. END IF
38. INVOKE Post\_Order\_Traversal(root.left)
39. INVOKE Post\_Order\_Traversal(root.right)
40. Print root.value
41. END FUNC
42. STOP

**CODE**

#include <iostream>

#include <cstdlib>

using namespace std;

typedef struct Node {

    int value;

    Node\* left;

    Node\* right;

} Node;

class BinarySearchTree {

    Node\* CreateNode(int value) {

        Node\* node = (Node\*) malloc(sizeof(Node));

        node->value = value;

        node->left = NULL;

        node->right = NULL;

        return node;

    }

    public:

    Node\* Root;

    BinarySearchTree(int value) {

        Node\* newNode = this->CreateNode(value);

        this->Root = newNode;

    }

    void Insert(int value) {

        Node\* newNode = this->CreateNode(value);

        Node\* root = this->Root;

        Node\* prevRoot = NULL;

        while (root != NULL) {

            prevRoot = root;

            if (newNode->value <= root->value) {

                root = root->left;

            } else {

                root = root->right;

            }

        }

        if (newNode->value <= prevRoot->value) {

            prevRoot->left = newNode;

        } else {

            prevRoot->right = newNode;

        }

    }

    void DisplaySubTree(Node\* node, int indent, int side) {

        for (int i=0; i<indent; i++) {

            cout << "    ";

        }

        if (side == 1) {

            cout << "L";

        } else if (side == 2) {

            cout << "R";

        }

        if (node->left == NULL) {

            cout << "|" << node->value << endl;

        } else {

            cout << "\\" << node->value << endl;

        }

        if (node->left != NULL) {

            this->DisplaySubTree(node->left, indent + 1, 1);

        }

        if (node->right != NULL) {

            this->DisplaySubTree(node->right, indent + 1, 2);

        }

    }

    void In\_Order\_Traversal(Node\* root) {

        if (root == NULL)

            return;

        In\_Order\_Traversal(root->left);

        cout<<root->value<<" ";

        In\_Order\_Traversal(root->right);

    }

    void Pre\_Order\_Traversal(Node\* root) {

        if (root == NULL)

            return;

        cout<<root->value<<" ";

        Pre\_Order\_Traversal(root->left);

        Pre\_Order\_Traversal(root->right);

    }

    void Post\_Order\_Traversal(Node\* root) {

        if (root == NULL)

            return;

        Post\_Order\_Traversal(root->left);

        Post\_Order\_Traversal(root->right);

        cout<<root->value<<" ";

    }

};

int main() {

    int choice, temp;

    cout << "Enter the root node value: ";

    cin >> temp;

    BinarySearchTree bst(temp);

    while (true) {

        cout << "\n---- Binary Search Tree (BST) ----" << endl;

        cout << "1. Insert" << endl;

        cout << "2. Inorder Traversal" << endl;

        cout << "3. Preorder Traversal" << endl;

        cout << "4. Postorder Traversal" << endl;

        cout << "5. Display BST" << endl;

        cout << "6. Exit" << endl;

        cout << "Enter Choice: ";

        cin >> choice;

        if (choice == 1) {

            cout << "Enter value to insert: ";

            cin >> temp;

            bst.Insert(temp);

        } else if (choice == 2) {

            cout << "Inorder => ";

            bst.In\_Order\_Traversal(bst.Root);

            cout << endl;

        } else if (choice == 3) {

            cout << "Preorder => ";

            bst.Pre\_Order\_Traversal(bst.Root);

            cout << endl;

        } else if (choice == 4) {

            cout << "Postorder => ";

            bst.Post\_Order\_Traversal(bst.Root);

            cout << endl;

        } else if (choice == 5) {

            bst.DisplaySubTree(bst.Root, 0, 0);

        } else if (choice == 6) {

            cout << "\n--------- Author ----------------" << endl;

            cout << "Ali Izzath Shazin" << endl;

            cout << "220071601028" << endl;

            cout << "B. Tech CSE A" << endl;

            break;

        } else {

            cout << "Invalid Option" << endl;

        }

    }

    return 0;}

**OUTPUT**

Enter the root node value: 10

---- Binary Search Tree (BST) ----

1. Insert

2. Inorder Traversal

3. Preorder Traversal

4. Postorder Traversal

5. Display BST

6. Exit

Enter Choice: 1

Enter value to insert: 5

---- Binary Search Tree (BST) ----

1. Insert

2. Inorder Traversal

3. Preorder Traversal

4. Postorder Traversal

5. Display BST

6. Exit

Enter Choice: 1

Enter value to insert: 15

---- Binary Search Tree (BST) ----

1. Insert

2. Inorder Traversal

3. Preorder Traversal

4. Postorder Traversal

5. Display BST

6. Exit

Enter Choice: 1

Enter value to insert: 8

---- Binary Search Tree (BST) ----

1. Insert

2. Inorder Traversal

3. Preorder Traversal

4. Postorder Traversal

5. Display BST

6. Exit

Enter Choice: 1

Enter value to insert: 12

---- Binary Search Tree (BST) ----

1. Insert

2. Inorder Traversal

3. Preorder Traversal

4. Postorder Traversal

5. Display BST

6. Exit

Enter Choice: 5

\10

L|5

R|8

R\15

L|12

---- Binary Search Tree (BST) ----

1. Insert

2. Inorder Traversal

3. Preorder Traversal

4. Postorder Traversal

5. Display BST

6. Exit

Enter Choice: 2

Inorder => 5 8 10 12 15

---- Binary Search Tree (BST) ----

1. Insert

2. Inorder Traversal

3. Preorder Traversal

4. Postorder Traversal

5. Display BST

6. Exit

Enter Choice: 3

Preorder => 10 5 8 15 12

---- Binary Search Tree (BST) ----

1. Insert

2. Inorder Traversal

3. Preorder Traversal

4. Postorder Traversal

5. Display BST

6. Exit

Enter Choice: 4

Postorder => 8 5 12 15 10

---- Binary Search Tree (BST) ----

1. Insert

2. Inorder Traversal

3. Preorder Traversal

4. Postorder Traversal

5. Display BST

6. Exit

Enter Choice: 6

--------- Author ----------------

Ali Izzath Shazin

220071601028

B. Tech CSE A