ASSIGNMENT 5 – BINARY SEARCH TREE

Code:

```
#include <iostream>
#include <stdlib.h>

using namespace std;

struct treeNode
{
    int data;
    treeNode *left;
    treeNode *right;
};

treeNode *FindMin(treeNode *node)
{
    if (node == NULL)
    {
        return NULL;
    }
    if (node->left)
    {
        return FindMin(node->left);
    }
    else
    {
        return node;
    }
}

treeNode *FindMax(treeNode *node)
{
```

```
    if (node == NULL)
    {
        return NULL;
    }
    if (node->right)
    {
        return FindMax(node->right);
    }
    else
    {
        return node;
    }
}

treeNode *Insert(treeNode *node, int data)
{
    if (node == NULL)
    {
        treeNode *temp = new treeNode;
        temp->data = data;
        temp->left = temp->right = NULL;
        return temp;
    }

    if (data > node->data)
    {
        node->right = Insert(node->right, data);
    }
    else if (data < node->data)
    {
        node->left = Insert(node->left, data);
    }
```

```
    return node;
}

treeNode *Delet(treeNode *node, int data)
{
    treeNode *temp;
    if (node == NULL)
    {
        cout << "Element not found";
    }
    else if (data < node->data)
    {
        node->left = Delet(node->left, data);
    }
    else if (data > node->data)
    {
        node->right = Delet(node->right, data);
    }
    else
    {
        if (node->right && node->left)
        {
            temp = FindMin(node->right);
            node->data = temp->data;
            node->right = Delet(node->right, temp->data);
        }
        else
        {
            temp = node;
            if (node->left == NULL)
            {
                node = node->right;
```

```
            }
            else if (node->right == NULL)
            {
                node = node->left;
            }
            free(temp);
        }
    }
    return node;
}

treeNode *Find(treeNode *node, int data)
{
    if (node == NULL)
    {
        return NULL;
    }
    else if (data > node->data)
    {
        return Find(node->right, data);
    }
    else if (data < node->data)
    {
        return Find(node->left, data);
    }
    else
    {
        return node;
    }
}

void Inorder(treeNode *node)
```

```
{
    if (node == NULL)
    {
        return;
    }
    Inorder(node->left);
    cout << node->data << " ";
    Inorder(node->right);
}

void Preorder(treeNode *node)
{
    if (node == NULL)
    {
        return;
    }
    cout << node->data << " ";
    Preorder(node->left);
    Preorder(node->right);
}

void Postorder(treeNode *node)
{
    if (node == NULL)
    {
        return;
    }
    Postorder(node->left);
    Postorder(node->right);
    cout << node->data << " ";
}
```

```
void DisplayLeafNodes(treeNode *node)
{
    if (node == NULL)
    {
        return;
    }
    if (node->left == NULL && node->right == NULL)
    {
        cout << node->data << " ";
    }
    DisplayLeafNodes(node->left);
    DisplayLeafNodes(node->right);
}


int main()
{
    treeNode *root = NULL, *temp;
    int ch, ch1, data1;
    while (1)
    {
        cout <<
"\n1.INSERT\n2.DELETE\n3.Inorder\n4.Preorder\n5.Postorder\n6.FindMin\n
7.FindMax\n8.Search\n9.Display Leaf Nodes\n10.Exit\n";
        cout << "Enter choice: ";
        cin >> ch;


        switch (ch)
        {
        case 1:
            cout << "\nEnter the number of elements to be inserted: ";
            cin >> ch1;
            for (int i = 0; i < ch1; i++)
            {
```

```
                cout << "\n\tPlease enter " << i + 1 << "th element:
";
                cin >> data1;
                root = Insert(root, data1);
            }
            cout << "\nElements in BST: ";
            Inorder(root);
            break;

        case 2:
            cout << "\nEnter element to be deleted: ";
            cin >> ch;
            root = Delet(root, ch);
            cout << "\nAfter deletion, elements in BST are: ";
            Inorder(root);
            break;

        case 3:
            cout << "Inorder traversal: ";
            Inorder(root);
            break;

        case 4:
            cout << "Preorder traversal: ";
            Preorder(root);
            break;

        case 5:
            cout << "Postorder traversal: ";
            Postorder(root);
            break;
```

```
        case 6:
            temp = FindMin(root);
            if (temp != NULL)
            {
                cout << "Smallest element in the tree is: " << temp-
>data;
            }
            else
            {
                cout << "The tree is empty.";
            }
            break;

        case 7:
            temp = FindMax(root);
            if (temp != NULL)
            {
                cout << "Largest element in the tree is: " << temp-
>data;
            }
            else
            {
                cout << "The tree is empty.";
            }
            break;

        case 8:
            cout << "\nEnter element to be searched: ";
            cin >> ch;
            temp = Find(root, ch);
            if (temp == NULL)
            {
                cout << "Element not found";
```

```
            }
            else
            {
                cout << "Element: " << temp->data << " is found\n";
            }
            break;

        case 9:
            cout << "Leaf nodes in the tree are: ";
            DisplayLeafNodes(root);
            cout << endl;
            break;

        case 10:
            exit(0);

        default:
            cout << "Enter correct choice: ";
            break;
        }
    }
    return 0;
}
```

```
OUTPUT:
1.INSERT
2.DELETE
3.Inorder
4.Preorder
5.Postorder
6.FindMin
7.FindMax
8.Search
9.Display Leaf Nodes
10.Exit
Enter choice: 1

Enter the number of elements to be inserted: 7

        Please enter 1th element: 5 3 7 2 4 6 8

        Please enter 2th element:
        Please enter 3th element:
        Please enter 4th element:
        Please enter 5th element:
        Please enter 6th element:
        Please enter 7th element:
Elements in BST: 2 3 4 5 6 7 8
1.INSERT
2.DELETE
3.Inorder
4.Preorder
5.Postorder
6.FindMin
7.FindMax
8.Search
```

```
9.Display Leaf Nodes
10.Exit
Enter choice: 2

Enter element to be deleted: 3

After deletion, elements in BST are: 2 4 5 6 7 8
1.INSERT
2.DELETE
3.Inorder
4.Preorder
5.Postorder
6.FindMin
7.FindMax
8.Search
9.Display Leaf Nodes
10.Exit
Enter choice: 3
Inorder traversal: 2 4 5 6 7 8
1.INSERT
2.DELETE
3.Inorder
4.Preorder
5.Postorder
6.FindMin
7.FindMax
8.Search
9.Display Leaf Nodes
10.Exit
Enter choice: 4
Preorder traversal: 5 4 2 7 6 8
1.INSERT
```

```
2.DELETE
3.Inorder
4.Preorder
5.Postorder
6.FindMin
7.FindMax
8.Search
9.Display Leaf Nodes
10.Exit
Enter choice: 5
Postorder traversal: 2 4 6 8 7 5
1.INSERT
2.DELETE
3.Inorder
4.Preorder
5.Postorder
6.FindMin
7.FindMax
8.Search
9.Display Leaf Nodes
10.Exit
Enter choice: 6
Smallest element in the tree is: 2
1.INSERT
2.DELETE
3.Inorder
4.Preorder
5.Postorder
6.FindMin
7.FindMax
8.Search
9.Display Leaf Nodes
```

```
10.Exit
Enter choice: 7
Largest element in the tree is: 8
1.INSERT
2.DELETE
3.Inorder
4.Preorder
5.Postorder
6.FindMin
7.FindMax
8.Search
9.Display Leaf Nodes
10.Exit
Enter choice: 8

Enter element to be searched: 2
Element: 2 is found

1.INSERT
2.DELETE
3.Inorder
4.Preorder
5.Postorder
6.FindMin
7.FindMax
8.Search
9.Display Leaf Nodes
10.Exit
Enter choice: 9
Leaf nodes in the tree are: 2 6 8

1.INSERT
```

2.DELETE

3.Inorder

4.Preorder

5.Postorder

6.FindMin

7.FindMax

8.Search

9.Display Leaf Nodes

10.Exit

Enter choice: 10