DSA Practical\codes\Practical1st.cpp

```cpp
#include <iostream>
using namespace std;
struct node
{
    int data;
    node *left, *right;
    node(int val)
    {
        data = val;
        left = NULL;
        right = NULL;
    }
};
void inorder(node *root)
{
    if (root == NULL)
    {
        return;
    }
    inorder(root->left);
    cout << root->data << " ";
    inorder(root->right);
}
void preorder(node *root)
{
    if (root == NULL)
    {
        return;
    }
    cout << root->data;
    cout << " ";
    inorder(root->left);
    inorder(root->right);
}
void postorder(node *root)
{
    if (root == NULL)
    {
        return;
    }
    inorder(root->left);
    inorder(root->right);
    cout << " ";
    cout << root->data;
}
node *search(node *root, int key)
{
    if (root == NULL)
```

```
    {
        return NULL;
    }
    else
    {
        if (root->data == key)
        {
            return root;
        }
        else if (root->data > key)
        {
            return search(root->left, key);
        }
        else
        {
            return search(root->right, key);
        }
    }
}
node *insert(node *root, int val)
{
    if (root == NULL)
    {
        root = new node(val);
    }
    else if (val > root->data)
    {
        root->right = insert(root->right, val);
    }
    else if (val < root->data)
    {
        root->left = insert(root->left, val);
    }
    else if (root->data == val)
    {
        cout << "Duplicate Value\n";
    }
    else
    {
        cout << "Invalid Entity";
    }
    return root;
}
node *inordersucc(node *root)
{
    node *curr = root;
    while (curr && curr->left != NULL)
    {
        curr = curr->left;
    }
```

```
    return curr;
}
node *deletenode(node *root, int del)
{
    if (del < root->data)
    {
        root->left = deletenode(root->left, del);
    }
    else if (del > root->data)
    {
        root->right = deletenode(root->right, del);
    }
    else
    {
        if (root->left == NULL)
        {
            node *temp = root->right;
            free(root);
            cout << "Node deleted"<<endl;
            return temp;
        }
        else if (root->right == NULL)
        {
            node *temp = root->left;
            free(root);
            cout << "Node deleted"<<endl;
            return temp;
        }
        node *temp = inordersucc(root->right);
        root->data = temp->data;
        root->right = deletenode(root->right, temp->data);
    }
    return root;
}
int main()
{
    int x, value;
    char ch, Y, N, n, y;
    node *root = NULL;
    cout << "Welcome to Binary Search Tree" << endl;
    while (true)
    {
        cout << "1]Insert node in tree" << endl
             << "2]Search for node in tree" << endl
             << "3]Traverse the tree" << endl
             << "4]Delete the node inside the tree" << endl
             << "5]To exit Program" << endl;
        cout << "Enter your choice:";
        cin >> x;
        switch (x)
```

```
        {
        case 1:
            cout << "Do you want to insert the node" << endl;
            while (true)
            {
                cout << "Y/y to continue or N/n to exit:";
                cin >> ch;
                if (ch == 'Y' || ch == 'y')
                {
                    cout << "Enter value to insert" << endl;
                    cin >> value;
                    root = insert(root, value);
                }
                else if (ch == 'N' || ch == 'n')
                {
                    cout << "Exiting...."<<endl;
                    break;
                }
                else
                {
                    cout << "Invalid Entity" << endl;
                }
            }
            break;
        case 2:
            cout << "Enter value to search for:" << endl;
            cin >> value;
            search(root, value);
            if (search(root, value) == NULL)
            {
                cout << "Element not found"<<endl;
            }
            else
            {
                cout << "Element found"<<endl;
            }
            break;
        case 3:
            cout << "Enter your choice:"<< endl;
            cout << "1]Inorder Traversal" << endl
                 << "2]Preorder Traversal " << endl
                 << "3]Postorder Traversal" << endl;
            cin >> x;
            switch (x)
            {
            case 1:
                inorder(root);
                cout<<endl;
                break;
            case 2:
```

```
                preorder(root);
                cout<<endl;
                break;
            case 3:
                postorder(root);
                cout<<endl;
                break;
            default:
                cout << "Invalid choice";
            }
            break;
        case 4:
            cout << "Enter value to delete" << endl;
            cin >> value;
            deletenode(root, value);
            break;
        case 5:
            cout << "Exiting....";
            break;
        default:
            cout << "Invalid choice";
            break;
        }
        if (x == 5)
        {
            break;
        }
    }
    return 0;
}

```