**Module 9**

#include <iostream>

#include <stack>

using namespace std;

class node **Topic 1**

{

public:

    node \*prev;

    int data;

    node \*next;

};

class linking

{

public:

    node \*h = NULL;

public:

    void create(int x)

    {

        node \*q = new node;

        node \*p;

        q->data = x;

        q->prev = NULL;

        q->next = NULL;

        if (h == NULL)

        {

            h = q;

        }

        else

        {

            p = h;

            while (1)

            {

                if (p->data >= x)

                {

                    if (p->prev == NULL)

                    {

                        p->prev = q;

                        break;

                    }

                    else

                    {

                        p = p->prev;

                    }

                }

                else

                {

                    if (p->next == NULL)

                    {

                        p->next = q;

                        break;

                    }

                    else

                    {

                        p = p->next;

                    }

                }

            }

        }

    }

    void createrecursivly(int n)

    {

        cout << "Enter the nodes : ";

        for (int i = 0; i < n; i++)

        {

            int x;

            cin >> x;

            create(x);

        }

    }

    void search(int x)

    {

        node \*ww = h;

        int c = 0;

        bool hh = true;

        if (ww != NULL)

        {

            while (ww != NULL)

            {

                if (ww->data == x)

                {

                    cout << x << " Data is found and depth is " << c - 1 << endl;

                    hh = false;

                    break;

                }

                else if (ww->data > x)

                {

                    ww = ww->prev;

                    c++;

                }

                else if (ww->data < x)

                {

                    ww = ww->next;

                    c++;

                }

            }

        }

        else if (ww == NULL)

            cout << "List is Empty" << endl;

        if (hh == true)

            cout << "Data not found" << endl;

    }

    void deletenode(int x)

    {

        node \*w = h;

        node \*temp = NULL;

        if (w != NULL)

        {

            while (w != NULL && w->data != x)

            {

                temp = w;

                if (w->data > x)

                    w = w->prev;

                else if (w->data < x)

                    w = w->next;

            }

            if (w != NULL)

            {

                if (w->prev == NULL || w->next == NULL)

                {

                    node \*U;

                    if (w->prev == NULL)

                        U = w->next;

                    else

                        U = w->prev;

                    if (w == temp->prev)

                        temp->prev = U;

                    else

                        temp->next = U;

                    if (U == NULL)

                        cout << "A leaf node " << w->data << " is deleted" << endl;

                    else

                        cout << "A node with one child " << w->data << " is deleted" << endl;

                    free(w);

                }

                else

                {

                    cout << "a node " << w->data << " two childs is deleted" << endl;

                    node \*a = NULL;

                    node \*b;

                    b = w->next;

                    while (b->prev != NULL)

                    {

                        a = b;

                        b = b->prev;

                    }

                    if (a != NULL)

                        a->prev = b->next;

                    else

                        w->next = b->next;

                    w->data = b->data;

                    free(w);

                }

            }

            else if (w == NULL)

            {

                cout << "Data not Found" << endl;

            }

        }

        else if (w == NULL)

        {

            cout << "List is Empty tooo" << endl;

        }

    }

    void display()

    {

        stack<node \*> st;

        node \*m = h;

        cout << "value "

             << "own address "

             << "left address "

             << "Right address " << endl;

        if (m == NULL)

            cout << "Tree is Empty" << endl;

        while (m != NULL || st.empty() != true)

        {

            if (m != NULL)

            {

                st.push(m);

                m = m->prev;

            }

            else

            {

                m = st.top();

                cout << m->data << " " << m << " " << m->prev << " " << m->next << endl;

                st.pop();

                m = m->next;

            }

        }

        cout << endl;

    }

};

// 70 35 97 88 44 32 90 15 30 60

// 10 7 5 8 15 11 18

/\*

     10

    /  \

   7    15

  / \   / \

  5  8 11 18

\*/

void menu()

{

    cout << "Binary Search Tree Menu\n1. Insert Node\n2. Search Node\n3. Delete Node\n4. Display Tree\n5. Exit\n"

         << endl;

    cout << "Enter your Option : ";

}

int main()

{

    linking test;

    int choice;

    while (choice != 5)

    {

        menu();

        cin >> choice;

        switch (choice)

        {

        case 1:

            int c2, n;

            cout << "Enter how you want to enter: 1. 1 by 1 2.recursive > ";

            cin >> c2;

            if (c2 == 1)

            {

                int x;

                cout << "Enter node : ";

                cin >> x;

                test.create(x);

                break;

            }

            else if (c2 == 2)

            {

                cout << "Enter how many nodes : ";

                cin >> n;

                test.createrecursivly(n);

                break;

            }

        case 2:

            cout << "Enter node You want to search : ";

            int x2;

            cin >> x2;

            test.search(x2);

            break;

        case 3:

            cout << "Enter node You want to Delete : ";

            int x3;

            cin >> x3;

            test.deletenode(x3);

            break;

        case 4:

            test.display();

            break;

        case 5:

            break;

        }

    }

}