Name : AARYAN BAIRAGI

Roll no: 27002

Class : SE IT-B

 Online C++ Compiler.

               Code, Compile, Run and Debug C++ program online.

Write your code in this editor and press "Run" button to compile and execute it.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

struct Node

{

    struct Node \*left,\*right;

    int data;

    bool leftthread;

    bool rightthread;

};

struct Node \*Insert(struct Node\*root, int key)

{

    struct Node \*node = root;

    struct Node \*par = NULL;

    while(node != NULL)

    {

        if(key == (node->data))

        {

            cout<<"Duplicate key"<<endl;

            return root;

        }

        par = node;

        if(key < (node->data))

        {

            if(node->leftthread == false)

            {

                node = node->left;

            }

            else

            break;

        }

        else

        {

            if(node->rightthread == false)

            {

                node = node->right;

            }

            else

            break;

        }

    }

    struct Node \*temp = new Node;

    temp->data = key;

    temp->leftthread = true;

    temp->rightthread = true;

    if(par == NULL)

    {

        root = temp;

        temp->left = NULL;

        temp->right = NULL;

    }

    else if (key < (par -> data))

    {

        temp -> left = par -> left;

        temp -> right = par;

        par -> leftthread = false;

        par -> left = temp;

    }

    else

    {

        temp -> left = par;

        temp -> right = par -> right;

        par -> rightthread = false;

        par -> right = temp;

    }

    return root;

}

struct Node \*InorderSuccessor(struct Node \*node)

{

    if(node->rightthread == true)

    {

        return (node->right);

    }

    else

    {

        node = node->right;

        while(node->leftthread == false)

        {

            node = node->left;

        }

    }

    return node;

}

void InorderTBT(struct Node \*root)

{

    if(root == NULL)

    {

        cout<<"Empty Tree"<<endl;

    }

    else

    {

        struct Node \*node = root;

        while(node->leftthread == false)

        {

            node = node->left;

        }

        while(node != NULL)

        {

            cout<<" "<<node->data;

            node = InorderSuccessor(node);

        }

    }

}

void PreorderTBT(struct Node \*root )

{

        struct Node \*node;

        if(root == NULL)

        {

                cout<<"Empty Tree "<<endl;

                return;

        }

        node = root;

        while(node != NULL)

        {

                cout<<" "<<node->data;

                if(node->leftthread == false)

                {

                        node=node->left;

                }

                else if(node->rightthread == false)

                {

                        node = node->right;

                }

                else

                {

                        while(node != NULL && node->rightthread == true)

                                node = node->right;

                        if(node != NULL)

                                node = node->right;

                }

        }

}

int main()

{

    struct Node \*root = NULL;

    int choice;

    while(1)

    {

        cout<<"1.Insert "<<endl;

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

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

        cout<<"Enter your choice : "<<endl;

        cin>>choice;

        switch(choice)

        {

            case 1:

            {

                cout<<"Enter the element you want to Insert : "<<endl;

                cin>>choice;

                root = Insert(root, choice);

            }

            break;

            case 2:

            {

                cout<<"Inorder Traversal Of TBT : ";

                InorderTBT(root);

                cout<<endl;

            }

            break;

            case 3:

            {

                cout<<"Preorder Traversal Of TBT : ";

                PreorderTBT(root);

                cout<<endl;

            }

            break;

            case 4:

            {

                exit(0);

            }

            break;

            default:

            {

                cout << "Invalid Choice." << endl;

            }

            break;

        }

    }

    return 0;

}