#include <iostream>

using namespace std;

class node

{

public:

    int data;

    node \*left, \*right;

    node()

    {

        data = 0;

        left = NULL;

        right = NULL;

    }

    node(int val)

    {

        data = val;

        left = NULL;

        right = NULL;

    }

};

class BST

{

public:

    node \*root;

    BST()

    {

        root = NULL;

    }

    void insert(int value)

    {

        node \*newNode = new node(value);

        if (root == NULL)

        {

            root = newNode;

            return;

        }

        node \*current = root;

        while (true)

        {

            if (value < current->data)

            {

                if (current->left == NULL)

                {

                    current->left = newNode;

                    break;

                }

                current = current->left;

            }

            else

            {

                if (current->right == NULL)

                {

                    current->right = newNode;

                    break;

                }

                current = current->right;

            }

        }

    }

    node \*checkifNodeExist(node \*rootnode, int val)

    {

        if (rootnode == NULL || rootnode->data == val)

            return rootnode;

        if (val > rootnode->data)

            return checkifNodeExist(rootnode->right, val);

        else

            return checkifNodeExist(rootnode->left, val);

    }

    node \*checkrightMinimum(node \*rootnode)

    {

        if (rootnode->left == NULL)

        {

            return rootnode;

        }

        return checkrightMinimum(rootnode->left);

    }

    void deletenode(int deleteval)

    {

        node \*deletenodeAddress = checkifNodeExist(root, deleteval);

        if (deletenodeAddress == NULL)

        {

            cout << "No Node with the given value is present in the tree\n";

        }

        else

        {

            node \*min = checkrightMinimum(deletenodeAddress->right);

            node \*temp = deletenodeAddress->right;

            deletenodeAddress->data = min->data;

            while (temp)

            {

                if (temp == min)

                {

                    deletenodeAddress->right = temp->right;

                    delete temp;

                    break;

                }

                if (temp->left == min)

                {

                    temp->left = min->right;

                    break;

                }

                temp = temp->left;

            }

        }

    }

    void inOrderTraversal(node \*root1)

    {

        if (root1 != nullptr)

        {

            inOrderTraversal(root1->left);

            cout << root1->data << " ";

            inOrderTraversal(root1->right);

        }

    }

};

int main()

{

    BST bst;

    bst.insert(50);

    bst.insert(30);

    bst.insert(20);

    bst.insert(40);

    bst.insert(70);

    bst.insert(60);

    bst.insert(80);

    cout<<"Before deletion\n";

    bst.inOrderTraversal(bst.root);

    bst.deletenode(50);

    bst.deletenode(60);

    cout<<"\nafter deletion\n";

    bst.inOrderTraversal(bst.root);

    return 0;

}

OUTPUT:

