#include <iostream>

#include <queue>

using namespace std;

struct Node

{

    int data;

    Node\* left, \* right;

};

Node\* newNode(int item)

{

    Node\* temp

        = (Node\*)malloc(sizeof(Node));

    temp->data = item;

    temp->left = temp->right = NULL;

    return temp;

}

/\*function to insert a new node with given key in BST \*/

Node\* insert(Node\* node, int key)

{

    // If the tree is empty, return a new node

    if (node == NULL)

        return newNode(key);

    // Otherwise, recur down the tree

    if (key < node->data)

        node->left = insert(node->left, key);

    else

        node->right = insert(node->right, key);

    // return the (unchanged) node pointer

    return node;

}

/\*return the node with minimum key value found in that tree \*/

Node\* minValueNode(Node\* node)

{

    Node\* current = node;

    /\* loop down to find the leftmost leaf \*/

    while (current && current->left != NULL)

        current = current->left;

    return current;

}

/\*function to delete the key and return the new root \*/

Node\* deleteNode(Node\* root, int key)

{

    if (root == NULL)

        return root;

    if (key < root->data)

        root->left = deleteNode(root->left, key);

    else if (key > root->data)

        root->right = deleteNode(root->right, key);

    else {

        // node has no child

        if (root->left==NULL and root->right==NULL)

            return NULL;

        // node with only one child or no child

        else if (root->left == NULL) {

            Node\* temp = root->right;

            free(root);

            return temp;

        }

        else if (root->right == NULL) {

            Node\* temp = root->left;

            free(root);

            return temp;

        }

        // node with two children: Get the inorder successor

        // (smallest in the right subtree)

        Node\* temp = minValueNode(root->right);

        root->data = temp->data;

        root->right = deleteNode(root->right, temp->data);

    }

    return root;

}

// Function to print tree nodes in Inorder fashion

void inorder(Node\* root)

{

    if (root != NULL)

    {

        inorder(root->left);

        cout << root->data <<" ";

        inorder(root->right);

    }

}

// Function to print tree nodes in level order fashion

void printLevelOrder(Node\* root)

{

    if (root == NULL)

        return;

    // Create an empty queue for level order traversal

    queue<Node\*> q;

    // Enqueue Root and initialize height

    q.push(root);

    while (q.empty() == false) {

        Node\* node = q.front();

        cout << node->data << " ";

        q.pop();

        if (node->left != NULL)

            q.push(node->left);

        if (node->right != NULL)

            q.push(node->right);

    }

}

// function to delete an element of given position from array

int delete\_from\_array(int arr[],int n,int x){

    int i;

    for(i=0;i<n;i++){

        if(arr[i]==x){

            break;

        }

    }

    if(i<n){

        n=n-1;

        for(int k=i;k<n;k++){

            arr[k]=arr[k+1];

        }

    }

    return n;

}

// function to print the elements of array

void display(int arr[],int n){

    for(int i=0;i<n;i++){

        cout<<arr[i]<<" ";

    }

}

// Driver program to test above function

int main()

{

    int n;

    cout<<"\nEnter the number of elements you want in array: ";

    cin>>n;

    int arr[n];

    cout<<"\nEnter the elements of the array: "<<endl;

    for(int i=0;i<n;i++){

        cin>>arr[i];

    }

    cout<<"\nThis is the array, you entered: "<<endl;

    cout<<"[" ;

    display(arr,n);

    cout<< "]"<<endl;

    //Displaying Binary Search Tree using Inorder Traversal

    Node\*root=NULL;

    cout<<"\nInorder Traversal of the given tree"<<endl;

    for(int i=0;i<n;i++){

        root=insert(root,arr[i]);

    }

    inorder(root);

    cout<<endl;

    //Displaying Binary Search Tree using level order Traversal

    cout<<"\nLevel order Traversal of the given tree"<<endl;

    printLevelOrder(root);

    cout<<endl;

    //deleting given data from Binary Search Tree

    int data1;

    cout<<"\nEnter the data to be deleted from BST:";

    cin>>data1;

    cout << "\nDeleting "<<data1<<endl;

    root = deleteNode(root, data1);

    cout << "\nInorder traversal of the modified tree: \n";

    inorder(root);

    cout<<endl;

    cout << "\nLevel order traversal of the modified tree: \n";

    printLevelOrder(root);

    cout<<endl;

    int x;

    cout<<"\nEnter the element of the array, you want to delete: ";

    cin>>x;

    cout<<"\nArray after deletion: "<<endl;

    delete\_from\_array(arr,n,x);

    display(arr,n-1);

}

/\*

BINARY SEARCH TREE

SEARCHING OPERATION:The space complexity of searching a node in a BST would be O(n).

INSERTION OPERATION:The space complexity of inserting a node in a BST would be O(n).

DELETION OPERTAION:The space complexity of this algorithm would be O(n).

with 'n' being the depth of the tree(number of nodes present in a tree)since at any

point of time maximum number of stack frames that could be present in memory is 'n'

ARRAY:

It has space complexity of O(n).\*/

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Ramneek\OneDrive\Desktop\DSA and AI-ML Lab\DSA> g++ .\binary\_search\_tree.cpp

PS C:\Users\Ramneek\OneDrive\Desktop\DSA and AI-ML Lab\DSA> .\a.exe

Enter the number of elements you want in array: 7

Enter the elements of the array:

50

30

20

40

70

60

80

This is the array, you entered:

[50 30 20 40 70 60 80 ]

Inorder Traversal of the given tree

20 30 40 50 60 70 80

Level order Traversal of the given tree

50 30 70 20 40 60 80

Enter the data to be deleted from BST:30

Deleting 30

Inorder traversal of the modified tree:

20 40 50 60 70 80

Level order traversal of the modified tree:

50 40 70 20 60 80

Enter the element of the array, you want to delete: 60

Array after deletion:

50 30 20 40 70 80

PS C:\Users\Ramneek\OneDrive\Desktop\DSA and AI-ML Lab\DSA>