```
/*
    CSLR41 Algorithms Laboratory
    Prajwal Sundar - Roll Number 106121092
    Program-1
    Print BFS and DFS orders of a given graph
*/

#include "bits/stdc++.h"
using namespace std;

class Graph
{
    public :

    int v; // number of vertices
    char * L; // labels of graphs
    int ** M; // adjacency matrix

    Graph() // constructor
    {
        cout << "Enter the number of vertices : ";
        cin >> v; // get vertices

        L = (char *) malloc(v * sizeof(char));
        cout << "Enter the labelling of your vertices : ";
        for (int i = 0; i < v; i++) cin >> L[i]; // get labelling

        M = (int **) malloc(v * sizeof(int *));
        for (int i = 0; i < v; i++)
        {
            M[i] = (int *) malloc(v * sizeof(int));
            for (int j = 0; j < v; j++) M[i][j] = 0;
        }
    }

    int getPos(char ch)
    {
        for (int i = 0; i < v; i++)
            if (L[i] == ch) return i; // position found
        return -1; // position not found
    }

    void addEdge()
    {
        char a, b;
        cin >> a >> b;

        int v1 = getPos(a), v2 = getPos(b); // endpoints of vertices
```

```
        M[v1][v2] = M[v2][v1] = 1; // set edge between vertices
    }

    // Print BFS order of the graph
    void BFS(char ch)
    {
        int visited[v]; // array of visited vertices
        memset(visited, 0, sizeof(visited));

        queue<int> Q; // queue
        int p = getPos(ch); // starting position

        Q.push(p); // enqueue current position
        visited[p] = 1; // current vertex is visited

        while (!Q.empty())
        {
            int x = Q.front(); // frontmost position
            Q.pop(); // dequeue
            cout << L[x] << " "; // print label

            for (int i = 0; i < v; i++)
            {
                if (!visited[i] && M[i][x])
                {
                    Q.push(i); // enqueue position
                    visited[i] = 1; // now position is visited
                }
            }
        }
    }

    // Print DFS order of the graph
    void DFS(char ch)
    {
        int visited[v]; // array of visited vertices
        memset(visited, 0, sizeof(visited));

        stack<int> S; // stack
        int p = getPos(ch); // starting position

        S.push(p); // push current position
        visited[p] = 1; // current vertex is visited

        while (!S.empty())
        {
            int x = S.top(); // topmost position
            S.pop(); // pop
```

```
            cout << L[x] << " "; // print label

            for (int i = v-1; i >= 0; i--)
            {
                if (!visited[i] && M[i][x])
                {
                    S.push(i); // push position
                    visited[i] = 1; // now position is visited
                }
            }
        }
    }
};

int main()
{
    cout << "Welcome to C++ BFS-DFS Displayer !" << endl << endl;

    Graph G; // graph object
    cout << endl;

    int e; // number of edges
    cout << "Enter the number of edges in your graph : ";
    cin >> e;

    cout << "In the next " << e << " pairs, enter end-vertices of each edge :-
" << endl;
    for (int i = 0; i < e; i++) G.addEdge(); // add edge

    char ch;
    cout << endl << "Enter the start vertex : ";
    cin >> ch;

    cout << "BFS Order : ";
    G.BFS(ch);

    cout << endl << "DFS Order : ";
    G.DFS(ch);

    cout << endl << endl << "Thank you for using C++ BFS-DFS Displayer. Bye
Bye !";
}
```

```cpp
/*
    CSLR41 Algorithms Laboratory
    Prajwal Sundar - Roll Number 106121092
    Program-2
    Form a binary tree using the given pre-order and in-order traversals
*/

#include "bits/stdc++.h"
using namespace std;

// Structure of a Binary Tree Node
struct Node
{
    int data; // data
    struct Node * left, * right; // left and right children

    Node(int n) // constructor
    {
        data = n; // set data
        left = right = NULL; // leaf node by default
    }
};

// Search an element in a sorted array between two given positions
int search(int * A, int n, int L, int U)
{
    for (int i = L; i <= U; i++)
        if (A[i] == n) return i; // element found
    return -1; // element not found
}

// Construct a binary tree using given pre and post orders
struct Node * formTree(int pre [], int in [], int L1, int U1, int L2, int U2)
{
    if (L1 > U1) return NULL;

    struct Node * root = new Node(pre[L1]);
    if (L1 == U1) return root;

    int P = search(in, pre[L1], L2, U2); // locate root in in-order
    root->left = formTree(pre, in, L1+1, P+L1-L2, L2, P-1); // form left
subtree
    root->right = formTree(pre, in, P+L1-L2+1, U1, P+1, U2); // form right
subtree

    return root;
}
```

```
// Print a binary tree with NULL children included
void print(struct Node * root)
{
    if (!root) cout << "N ";
    else
    {
        cout << root->data << " "; // root
        print(root->left); // recursion in left subtree
        print(root->right); // recursion in right subtree
    }
}

int main()
{
    cout << "Welcome to C++ Pre-In Order Binary Tree Creator !" << endl <<
endl;

    int n; // number of elements
    cout << "Enter the number of elements in your tree : ";
    cin >> n;

    int pre[n]; // pre-order array
    cout << "Enter the pre-order traversal of your array : ";
    for (int i = 0; i < n; i++) cin >> pre[i];

    int in[n]; // in-order array
    cout << "Enter the in-order traversal of Syour array : ";
    for (int i = 0; i < n; i++) cin >> in[i];

    struct Node * root = formTree(pre, in, 0, n-1, 0, n-1);
    cout << endl << "Binary Tree was formed successfully ! The tree is : " <<
endl;
    print(root);

    cout << endl << endl << "Thank you for using C++ Pre-In Binary Tree
Creator. Bye Bye !";
}
```

```
/*
    CSLR41 Algorithms Laboratory
    Prajwal Sundar - Roll Number 106121092
    Program-3
    Form a binary tree using the given pre-order and post-order traversals
*/

#include "bits/stdc++.h"
using namespace std;

// Structure of a Binary Tree Node
struct Node
{
    char data; // data
    struct Node * left, * right; // left and right children

    Node(char n) // constructor
    {
        data = n; // set data
        left = right = NULL; // leaf node by default
    }
};

// Search an element in a sorted array between two given positions
int search(char * A, char n, int L, int U)
{
    for (int i = L; i <= U; i++)
        if (A[i] == n) return i; // element found
    return -1; // element not found
}

// Construct a binary tree using given pre and post orders
struct Node * formTree(char pre [], char post [], int L1, int U1, int L2, int
U2)
{
    if (L1 > U1) return NULL;

    struct Node * root = new Node(pre[L1]);
    if (L1 == U1) return root;

    if (pre[L1+1] == post[U2-1]) // full binary tree not possible, make it
left only
        root->left = formTree(pre, post, L1+1, U1, L2, U2-1);

    else
    {
        int P1 = search(post, pre[L1+1], L2, U2);
        int P2 = search(pre, post[U2-1], L1, U1);
```

```
        root->left = formTree(pre, post, L1+1, P2-1, L2, P1); // form left
subtree
        root->right = formTree(pre, post, P2, U1, P1+1, U2-1); // form right
subtree
    }

    return root;
}

// Print a binary tree with NULL children included
void print(struct Node * root)
{
    if (!root) cout << "N ";
    else
    {
        cout << root->data << " "; // root
        print(root->left); // recursion in left subtree
        print(root->right); // recursion in right subtree
    }
}

int main()
{
    cout << "Welcome to C++ Pre-Post Order Binary Tree Creator !" << endl <<
endl;

    int n; // number of elements
    cout << "Enter the number of elements in your tree : ";
    cin >> n;

    char pre[n]; // pre-order array
    cout << "Enter the pre-order traversal of your array : ";
    for (int i = 0; i < n; i++) cin >> pre[i];

    char post[n]; // post-order array
    cout << "Enter the post-order traversal of your array : ";
    for (int i = 0; i < n; i++) cin >> post[i];

    struct Node * root = formTree(pre, post, 0, n-1, 0, n-1);
    cout << endl << "Binary Tree was formed successfully ! The tree is : " <<
endl;
    print(root);

    cout << endl << endl << "Thank you for using C++ Pre-Post Binary Tree
Creator. Bye Bye !";
}
```