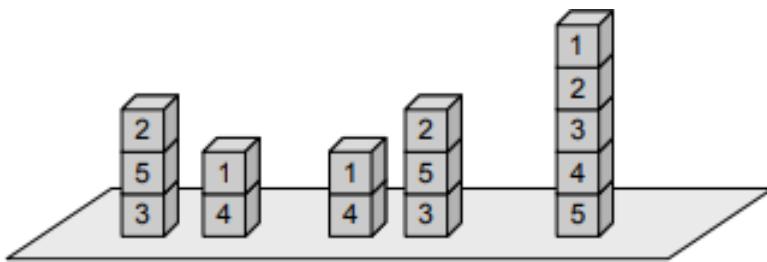


Assignment No: 1
Prn:2020BTECS00090
Name:Vishal Chauhan
Sub:AI-ML Lab

1. The block's world is a well-known toy domain used in AI for planning problems related to robots. It consists of a set of blocks of various size, shape, and color, possibly identified by a letter or by a number, and placed on a surface (e.g., on a table); the blocks can be stacked into one or more piles, possibly with some constraints (depending, e.g., on their size); the goal is to form a target set of piles starting from a given initial configuration, by moving one block at a time, either to the table or atop another pile of blocks, and only if it is not under another block. Consider a simple version of this problem, in which there are five blocks with identical shape and size, numbered from 1 to 5.

The figure below shows two possible configurations of such blocks; note that the relative position of the piles does not matter, thus the configuration on the left is equivalent to the one in the middle. Every block can be either on the table (there are no constraints on the number of piles) or atop any other block. The goal is to stack the blocks in a single pile, in the order shown in the rightmost configuration in the figure, starting from any given set of piles, by moving the smallest possible number of blocks. Only blocks at the top of the current piles can be moved, and can be placed only on the table or a top another pile.



Formulate the above version of the block's world as a search problem, by precisely defining the state space, the initial state, the goal test, the set of possible actions and the path cost (based on cost of moving blocks).

Answer:-

The state space will be made up of set of distinct arrangements of the five blocks. The current state can be represented using the specific arrangement of the blocks in each pile. The possible or legal actions would be to move a single block from the top of the pile or ground and put it either on the top of the another pile or on the ground.

Here the goal state would be defined or said to be reached when in minimum number of moves we reach the target configuration and a path cost would be associated with it and the path cost will be given by the number of moves made to reach the goal state. Thus fro each move the cost would be equal to 1.

2. You start with the sequence ABABAECCEC, or in general any sequence made from A, B, C, and E. You can transform this sequence using the following equalities: $AC = E$, $AB = BC$, $BB = E$, and $E * x = x$ for any x . For example, ABBC can be transformed into AEC, and then AC, and then E. Your goal is to produce the sequence E.

Date _____
Page _____

Q2. You start with the sequence ABABAECCEC or in general any sequence made from A, B, C and E. You can transform this sequence made from A, B, C and E. You can transform this sequence using the following equalities: AC = E, AB = BC, BB = E and $E^*x = x$ for any x. For example ABBC can be transformed into AEC, and then AC and then E. Your goal is to produce the sequence E.

→ Answer :-

Approach 1:-

ABABAEFCCGEC

Step 1: Converting the AB to BC we get

ABBCAECCEC

Step 2: Converting the BB to E

AECAECCEC

Step 3: Converting AE, EC, EC to corresponding A, E, E

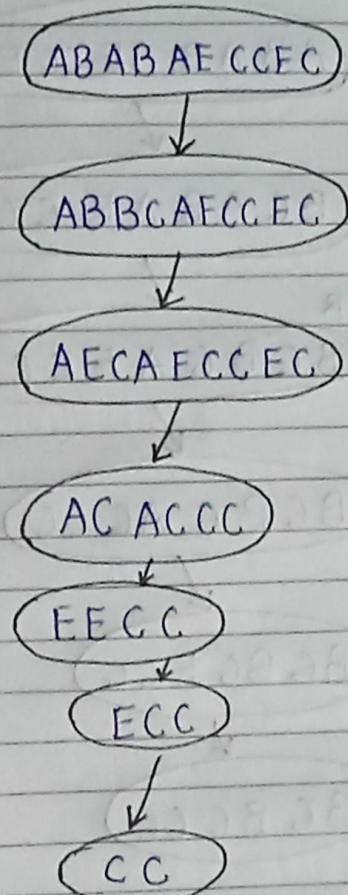
AACEC ACACCC

Step 4: Converting AC to E

AEEE EECC

Date _____
Page _____

Approach I:-



Finally we get CC as the reduced
which is not our goal state hence
the approach is discarded.

Date _____
Page _____

Step 5:- Converting EC to C
ECC

Step 6:- Converting EC to C
C.C

So we can conclude as this is the least possible reduced by this method so we can understand that this method doesn't give solution so we need to try an alternative method.

Approach 2:-

ABABAECCEC

Step 1:- Converting AB to BC
BCABAECCEC

Step 2:- Converting AB to BC
BCBCAECCEC

Step 3:- Converting EC to C.
BCBBCACCC

~~But we cannot~~

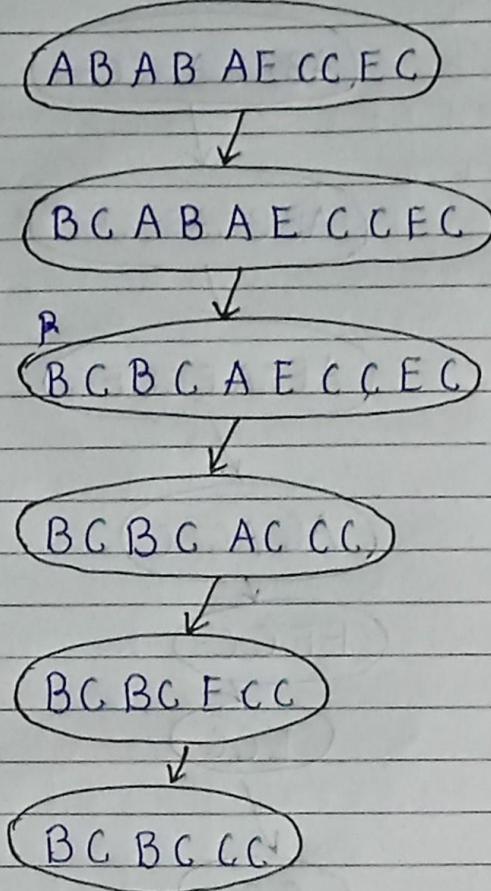
Step 4:- Converting AC to E.
BCBBCCEC

Step 5:- Converting EC to C

BCBBCCC

We cannot further reduce the string

Approach 2:-

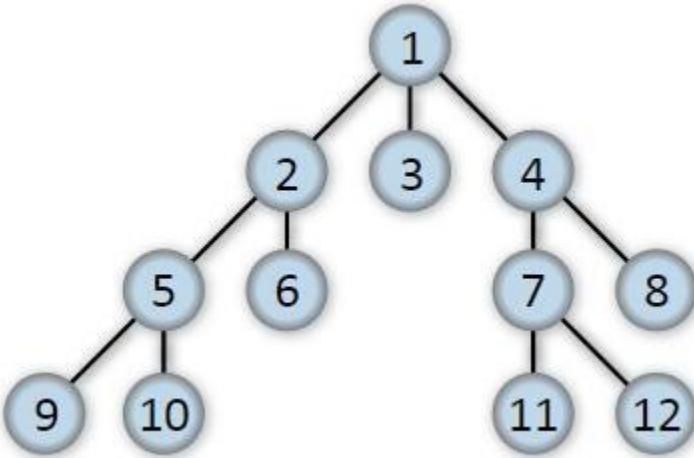


Finally we get BCBGCC as the reduced string and we cannot further reduce it hence this approach is also discarded as it is not equal to E which is our goal state.

Date _____
Page _____

Similarly we can check for every possible approach and end we conclude there is no solution for this question.

3. For the tree given below:



If starting state is 1 and goal state is 7, implement BFS and DFS search strategies for above tree. Which searching strategy will be best for this problem? Justify your answer with proper explanation.

DFS Output:-

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n,m;
    cout<<"Enter the number of nodes and number of edges in the graph: "<<endl;
    cin>>n>>m;
    int key;
    cout<<"Enter the element you want to find: "<<endl;
    cin>>key;
    vector<int> adj[n+1];
    vector<int> visited(n+1,0);
    for(int i=0;i<m;i++){
        int a,b;
        cin>>a>>b;
        adj[a].push_back(b);
        adj[b].push_back(a);
    }

    cout<<"Enter the links between the nodes"<<endl;
    stack<int> q;
    q.push(1);
    visited[1]=1;
    while(!q.empty()){
        int data = q.top();
        cout<<data<<" ";
        if(data==key){
            break;
        }
        q.pop();
        reverse(adj[data].begin(),adj[data].end());
        for(auto it:adj[data]){
            if(!visited[it]){
                visited[it] = 1;
                q.push(it);
            }
        }
    }
}
```

```
        q.push(it);
    }
}

}
return 0;
}
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "AI ASSIGNMENTS" containing files: .vscode, Question_1.cpp, Question_3_BFS.cpp, Question_3_DFS.cpp, Question_3_DFS.exe, and Question_3.exe.
- Code Editor:** The active file is Question_3_BFS.cpp. The code is as follows:

```
C++ Question_3_BFS.cpp
C++ Question_3_DFS.cpp X

C++ Question_3_DFS.cpp > main()
7   cin>>n>>m;
8   int key;
9   cout<<"Enter the element you want to find: "<<endl;
10  cin>>key;

7
1 2
1 3
1 4
2 5
2 6
5 9
5 10
4 7
4 8
7 11
7 12

Enter the links between the nodes
1 2 5 9 10 6 3 4 7
```

- Terminal:** Shows the command PS D:\D Drive Data\Vyankatesh Bura\Engineering B Tech\Books\Engineering\Computer
- Status Bar:** Shows the current position as Line 11, Column 26, with other settings like Spaces: 4, UTF-8, CRLF, C++, Go Live, Win32, Prettier, and a timestamp of 2:26 PM 8/24/2022.

BFS Output:-

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n,m;
    cout<<"Enter the number of nodes and number of edges in the graph: "<<endl;
    cin>>n>>m;
    int key;
    cout<<"Enter the element you want to find: "<<endl;
    cin>>key;
    vector<int> adj[n+1];
    vector<int> visited(n+1,0);

    cout<<"Enter the links between the nodes"<<endl;
    for(int i=0;i<m;i++){
        int a,b;
        cin>>a>>b;
        adj[a].push_back(b);
        adj[b].push_back(a);
    }

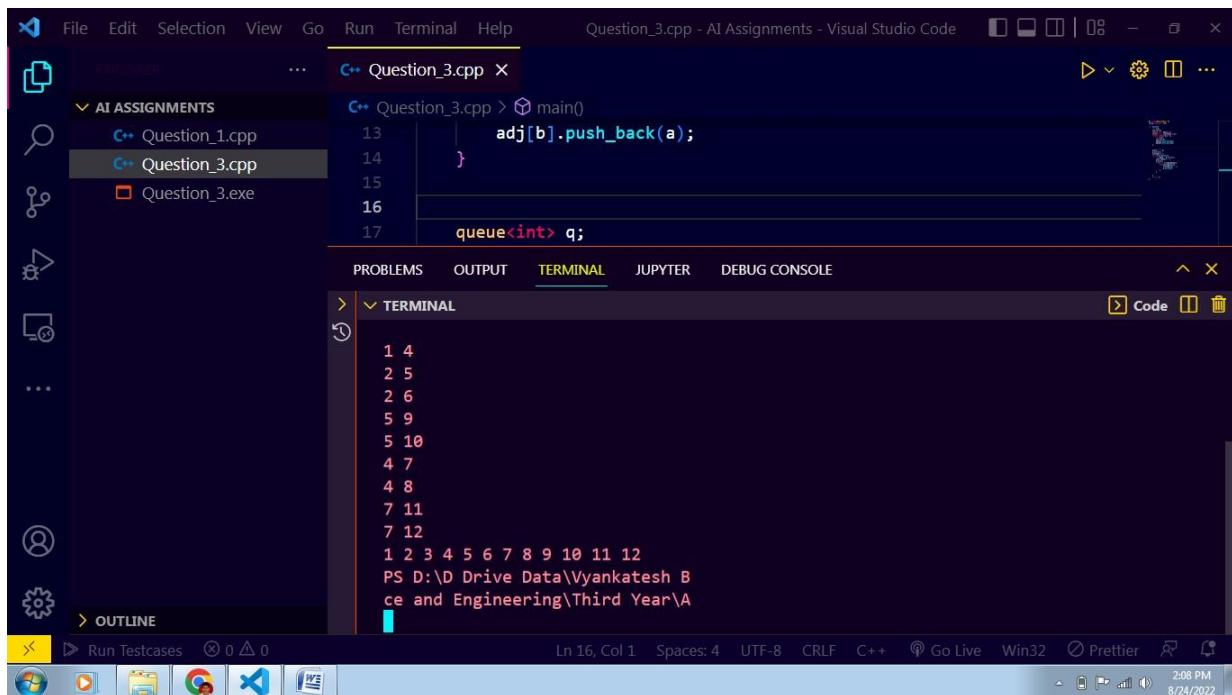
    queue<int> q;
    q.push(1);
    visited[1]=1;
    while(!q.empty()){

        int node=q.front();
        cout<<"Visited Node: "<<node<<endl;
        q.pop();

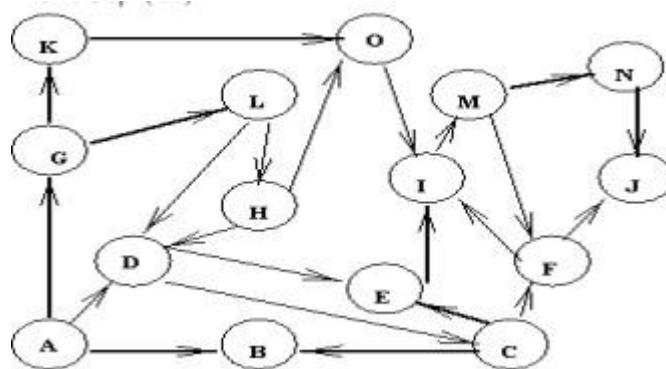
        for(int i=0;i<adj[node].size();i++){
            int neighbor=adj[node][i];
            if(visited[neighbor]==0){
                q.push(neighbor);
                visited[neighbor]=1;
            }
        }
    }
}
```

```
int data = q.front();
cout<<data<<" ";
if(data==key){
    break;
}
q.pop();
for(auto it:adj[data]){
    if(!visited[it]){
        visited[it] = 1;
        q.push(it);
    }
}

return 0;
}
```



4. Consider the traffic map of certain city is given below



Identify different paths using

- a. BFS with start state A and goal state N

CPP Implementation:-

```
#include <bits/stdc++.h>
using namespace std;

void printpath(vector<int>& path)
{
    char ch='A';
    map<int,char>mp;

    for(int i=0;i<15;i++)
    {
        mp[i]=ch;
        ch++;
    }

    int size = path.size();
    for (int i = 0; i < size; i++)
        cout << mp[path[i]] << " ";
    cout << endl;
}

int isNotVisited(int x, vector<int>& path)
{
    int size = path.size();
    for (int i = 0; i < size; i++)
        if (path[i] == x)
            return 0;
    return 1;
}

void findpaths(vector<vector<int> >&g, int src,int dst, int v)
{
    queue<vector<int> > q;
    vector<int> path;
```

```
path.push_back(src);
q.push(path);
while (!q.empty()) {
    path = q.front();
    q.pop();
    int last = path[path.size() - 1];

    if (last == dst)
        printpath(path);

    for (int i = 0; i < g[last].size(); i++) {
        if (isNotVisited(g[last][i], path)) {
            vector<int> newpath(path);
            newpath.push_back(g[last][i]);
            q.push(newpath);
        }
    }
}

int main()
{
    int n=16;
    vector<vector<int>> g(n+1);
    // number of vertices

    int m=23;

    for(int i=0;i<m;i++)
    {
        char c1,c2;
        cin>>c1>>c2;

        int u=c1-'A';
        int v=c2-'A';

        // cout<<u<<" "<<v<<endl;
        g[u].push_back(v);

    }

    int src = 0, dst = 13;
    char sr = 'A'+0;
    char ds = 'A'+13;
    cout << "path from src " << sr
        << " to dst " << ds << " are \n";

    // function for finding the paths
    findpaths(g, src, dst, n);

    return 0;
}
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder structure with files like 'G_L', 'L_H', 'K_O', and 'I_N'. It also displays 'Expected Output' and 'Received Output' sections.
- Code Editor:** The active file is 'Question_no_4.cpp'. The code implements DFS to find paths from state A to state N. It includes a function 'findpaths' and prints the path found.
- Terminal:** Shows the command 'cd "/mnt/c/Users/vyankateshbura/Desktop/Assignment 4"' followed by the output of the program.
- Status Bar:** Displays file path ('Question_no_4.cpp'), line ('Ln 91, Col 6'), and date ('8/26/2022').

b. DFS with start state A and goal state N

Cpp Implementation:-

```
#include <bits/stdc++.h>
using namespace std;

void printpath(vector<int>& path)
{
    char ch='A';
    map<int,char>mp;

    for(int i=0;i<15;i++)
    {
        mp[i]=ch;
        ch++;
    }

    int size = path.size();
    for (int i = 0; i < size; i++)
        cout << mp[path[i]] << " ";
    cout << endl;
}

int isNotVisited(int x, vector<int>& path)
{
    int size = path.size();
    for (int i = 0; i < size; i++)
        if (path[i] == x)
            return 0;
    return 1;
}
```

```
void findpaths(vector<vector<int> >&g, int src,int dst, int v)
{
    stack<vector<int> > q;

    vector<int> path;
    path.push_back(src);
    q.push(path);
    while (!q.empty()) {
        path = q.top();
        q.pop();
        int last = path[path.size() - 1];

        if (last == dst)
            printpath(path);

        for (int i = 0; i < g[last].size(); i++) {
            if (isNotVisited(g[last][i], path)) {
                vector<int> newpath(path);
                newpath.push_back(g[last][i]);
                q.push(newpath);
            }
        }
    }
}

int main()
{
    int n=16;
    vector<vector<int> > g(n+1);
    // number of vertices

    int m=23;
    for(int i=0;i<m;i++)
    {
        char c1,c2;
        cin>>c1>>c2;

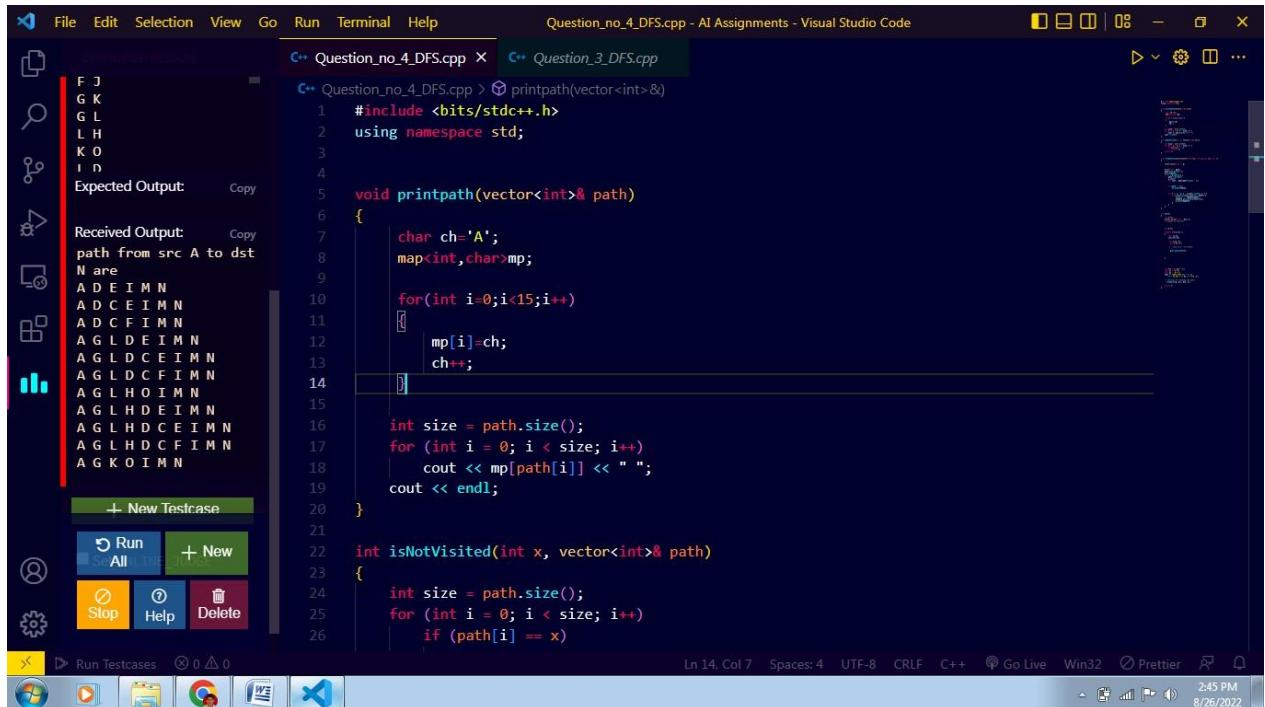
        int u=c1-'A';
        int v=c2-'A';

        // cout<<u<<" "<<v<<endl;
        g[u].push_back(v);

    }

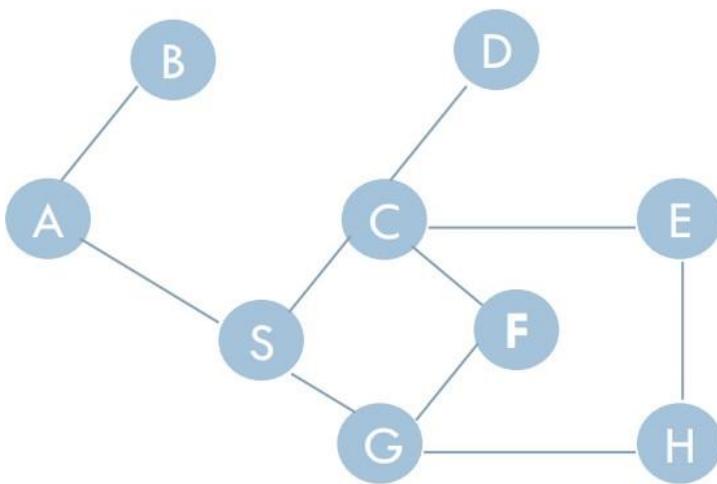
    int src = 0, dst = 13;
    char sr = 'A'+0;
    char ds = 'A'+13;
    cout << "path from src " << sr
```

```
<< " to dst " << ds << " are \n";\n\n    // function for finding the paths\n    findpaths(g, src, dst, n);\n\n    return 0;\n}
```



```
File Edit Selection View Go Run Terminal Help Question_no_4_DFS.cpp - AI Assignments - Visual Studio Code\n\nC++ Question_no_4_DFS.cpp X C++ Question_3_DFS.cpp\n\nC++ Question_no_4_DFS.cpp > printpath(vector<int>& path)\n1 #include <bits/stdc++.h>\n2 using namespace std;\n3\n4\n5 void printpath(vector<int>& path)\n6 {\n7     char ch='A';\n8     map<int,char>mp;\n9\n10    for(int i=0;i<15;i++)\n11    {\n12        mp[i]=ch;\n13        ch++;\n14    }\n15\n16    int size = path.size();\n17    for (int i = 0; i < size; i++)\n18        cout << mp[path[i]] << " ";\n19    cout << endl;\n20}\n21\n22 int isNotVisited(int x, vector<int>& path)\n23 {\n24     int size = path.size();\n25     for (int i = 0; i < size; i++)\n26         if (path[i] == x)\n\nLn 14, Col 7 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier 2:45 PM 8/26/2022
```

5. For the graph given below



Implement BFS and DFS technique to identify a path from start node A to goal node E.

Answer:

BFS Traversal:

Implementation:

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n,m;
    cout<<"Enter the number of nodes and number of edges in the graph: "<<endl;
    cin>>n>>m;
    cout<<"Enter the starting node of BFS Traversal:"<<endl;
    int st;
    cin>>st;
    int key;
    cout<<"Enter the element you want to find: "<<endl;
    cin>>key;
    vector<int> adj[n+1];
    vector<int> visited(n+1,0);

    cout<<"Enter the links between the nodes"<<endl;
    for(int i=0;i<m;i++){
        char a,b;
        cin>>a>>b;
        int var1,var2;
        var1 = a-'A';
        var2 = b - 'A';
        adj[var1].push_back(var2);
        adj[var2].push_back(var1);
    }

    queue<int> q;
    q.push(1);
    visited[1]=1;
    while(!q.empty()){
        int data = q.front();
        char ch = data +'A';
        cout<<ch<<" ";
        if(data==key){
            break;
        }
        q.pop();
        for(auto it:adj[data]){
            if(!visited[it]){
                visited[it] = 1;
                q.push(it);
            }
        }
    }
    return 0;
}
```

Output:

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal window displays the execution of a C++ program named 'Question_no_6.cpp'. The program prompts the user for the number of nodes and edges in the graph, which are entered as 26 and 10 respectively. It then asks for the starting node of BFS Traversal, which is 0. Next, it asks for the element to find, which is 4. Finally, it asks for the links between the nodes, with the input being: A B, A S, S C, C D, C E, C F, S G, G H, H E, G F, B A S C G D E. The terminal window also shows the current working directory as 'D:\D Drive Data\Vyankatesh Bura\Engineering B Tech\Books\Engineering\Computer Science and Engineering\Third Year\Academics\AI Assignments'.

DFS Traversal:

Implementation:

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    int n,m;
    // cout<<"Enter the number of nodes and number of edges in the graph: "<<endl;
    cin>>n>>m;
    // cout<<"Enter the starting node of BFS Traversal:"<<endl;
    int st;
    cin>>st;
    int key;
    // cout<<"Enter the element you want to find: "<<endl;
    cin>>key;
    vector<int> adj[n+1];
    vector<int> visited(n+1,0);

    // cout<<"Enter the links between the nodes"<<endl;
    for(int i=0;i<m;i++){
        char a,b;
        cin>>a>>b;
        int var1,var2;
        var1 = a-'A';
        var2 = b - 'A';
        adj[var1].push_back(var2);
        adj[var2].push_back(var1);
    }
}
```

```
stack<int> q;
q.push(1);
visited[1]=1;
while(!q.empty()){
    int data = q.top();
    char ch = data + 'A';
    cout<<ch<< " ";
    if(data==key){
        break;
    }
    q.pop();
    for(auto it:adj[data]){
        if(!visited[it]){
            visited[it] = 1;
            q.push(it);
        }
    }
}
return 0;
}
```

Output:

The screenshot shows the Visual Studio Code interface with three tabs open: `Question_no_6B.cpp`, `Question_no_6B.cpp` (active), and `Question_3_BFS.cpp`. The active tab displays the following C++ code for a Breadth-First Search (BFS) traversal:

```
int main(){
    int n,m;
    // cout<<"Enter the number of nodes and number of edges in the graph: "<<endl;
    cin>>n>>m;
    // cout<<"Enter the starting node of BFS Traversal:<<endl;
    int st;
    cin>>st;
    int key;
    // cout<<"Enter the element you want to find: "<<endl;
    cin>>key;
    vector<int> adj[n+1];
    vector<int> visited(n+1,0);

    // cout<<"Enter the Links between the nodes"<<endl;
    for(int i=0;i<m;i++){
        char a,b;
        cin>>a>>b;
        int var1,var2;
        var1 = a - 'A';
        var2 = b - 'A';
        adj[var1].push_back(var2);
        adj[var2].push_back(var1);
    }

    stack<int> q;
    q.push(1);
    visited[1]=1;
    while(!q.empty()){
        int data = q.top();
        char ch = data + 'A';
        cout<<ch<< " ";
        if(data==key){
            break;
        }
        q.pop();
        for(auto it:adj[data]){
            if(!visited[it]){
                visited[it] = 1;
                q.push(it);
            }
        }
    }
}
return 0;
}
```

The left sidebar shows a test case with 1 Passed result in 85ms. The input is given as:

```
26 10
0 4
A B
A S
S C
S G
C D
C E
C F
G F
G H
H E
```

The expected output is:

```
B A S G H E
```

The received output is:

```
B A S G H E
```

At the bottom, there are buttons for Run All, Stop, Help, and Delete.