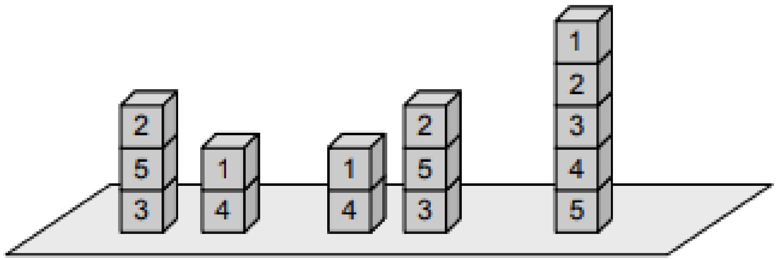
Vyankatesh Sanjay Bura
2020BTECS00035

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 another 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.

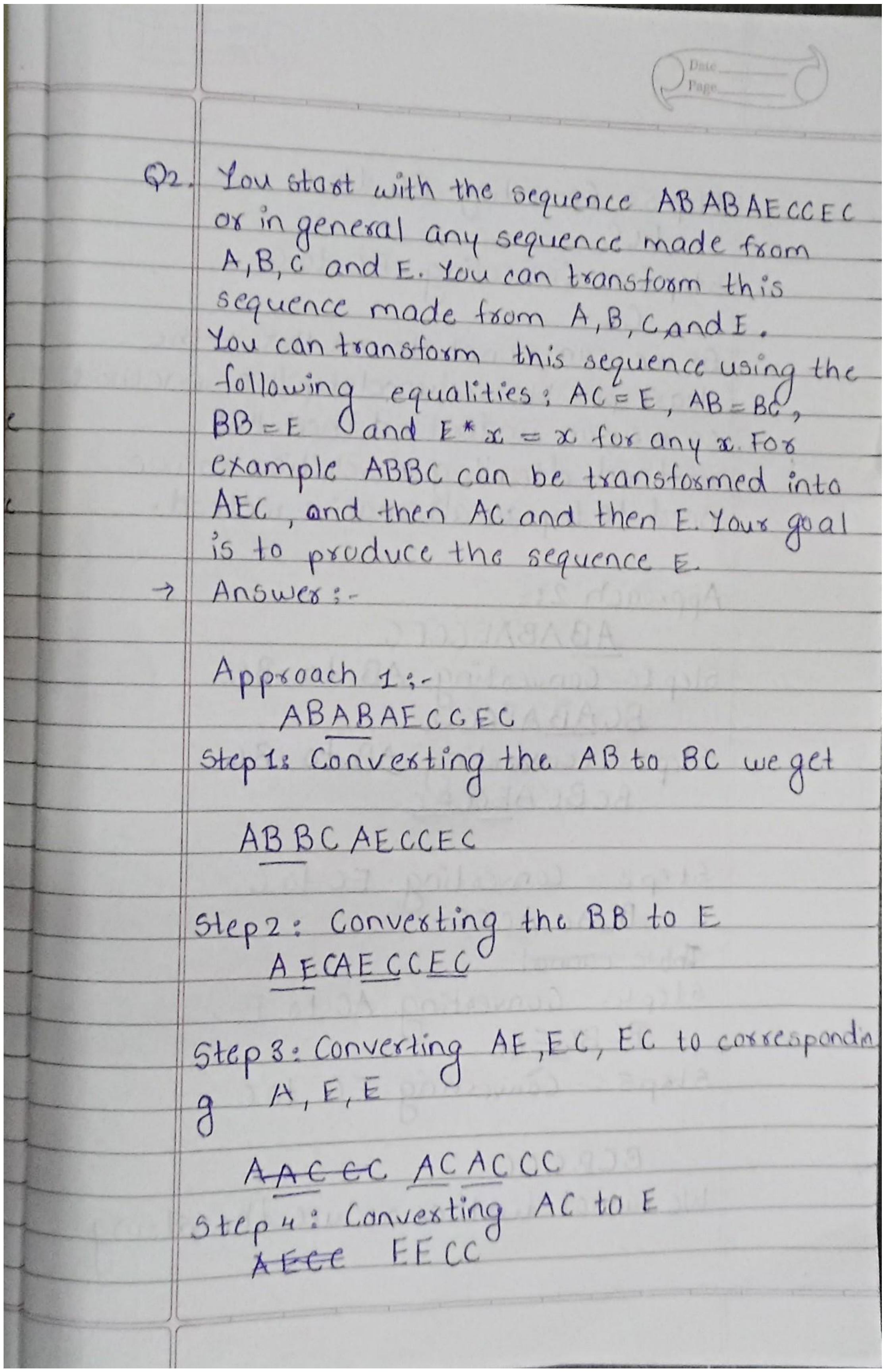


   Formulate the above version of the block's world as a search problem, by precisely defining the state space, the initial state, the goal test, the set of possible actions and the path cost (based on cost of moving blocks).

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.

Vyankatesh Sanjay Bura
2020BTECS00035



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:-

AB$\underline{AB}$AECCEC

Step 1: Converting the AB to BC we get

$\underline{AB}$BCAECCEC

Step 2: Converting the BB to E

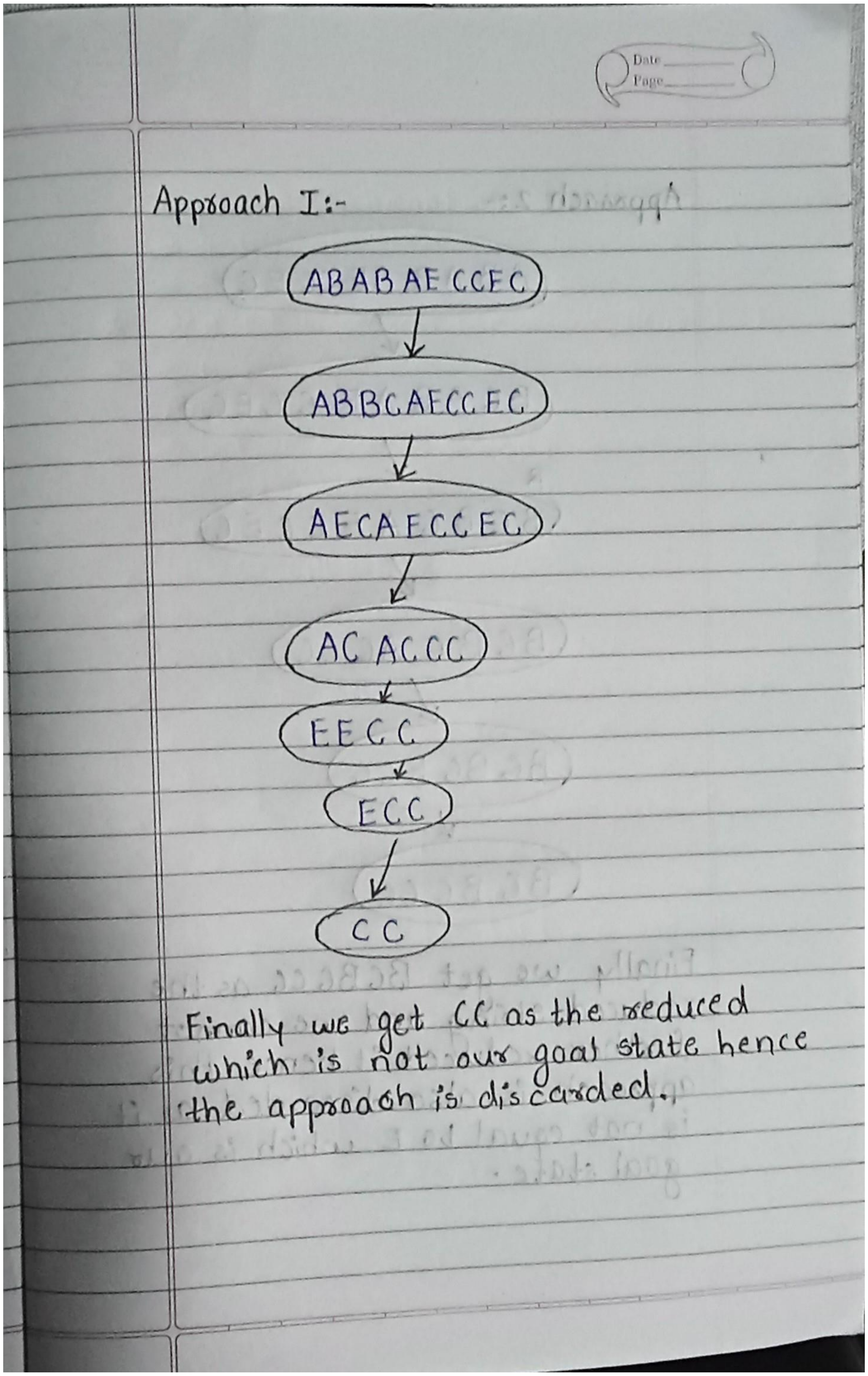
$\underline{A}$ $\underline{E}$CA$\underline{E}$ $\underline{C}$C$\underline{EC}$

Step 3: Converting AE, EC, EC to correspondin g A, E, E

~~AACEC~~ $\underline{AC}$ $\underline{AC}$ CC

Step 4: Converting AC to E

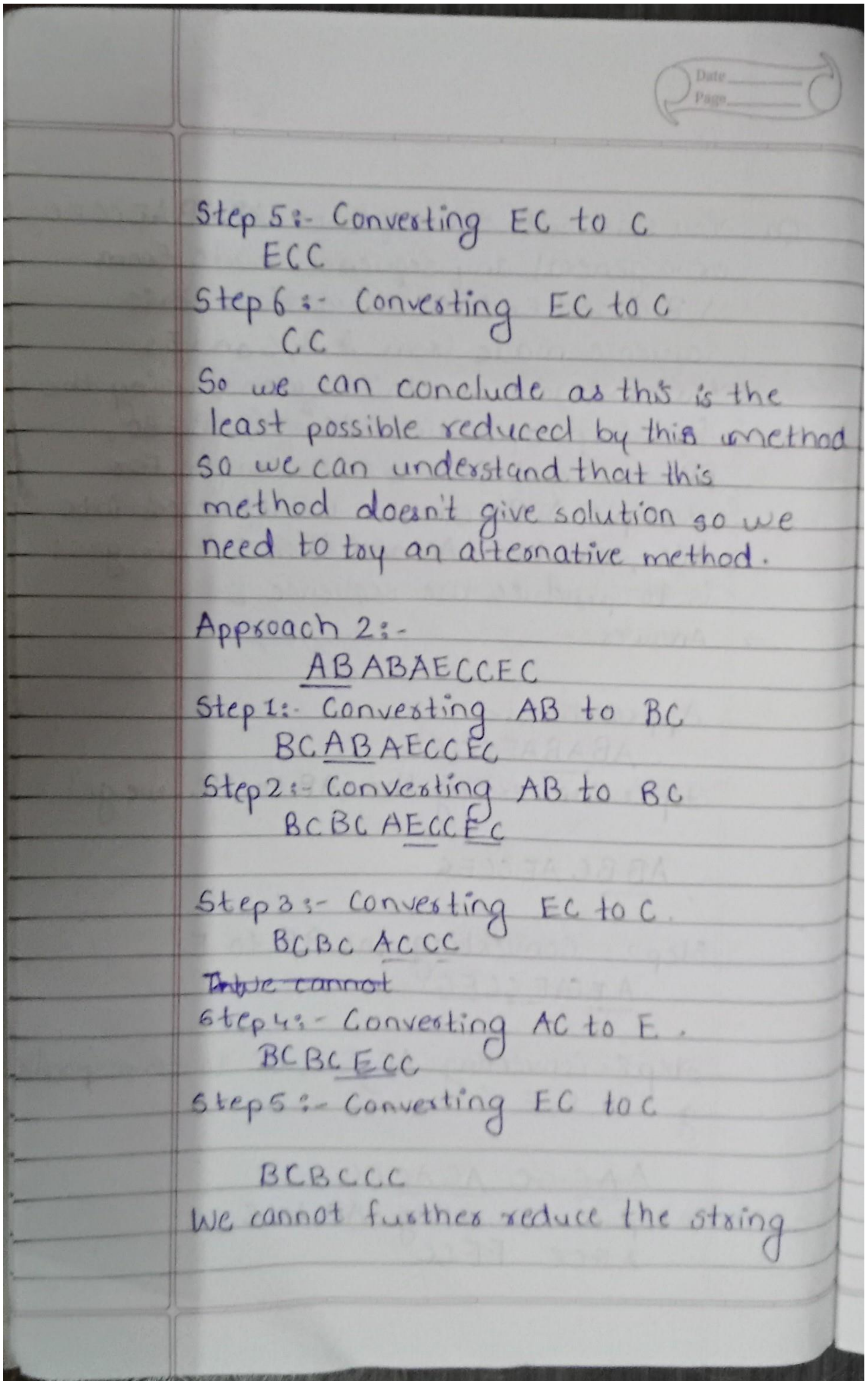
~~AECC~~ EECC



Approach I:-

$$\begin{tikzcd}
\text{ABABAECCEC} \arrow[d] \\
\text{ABBCAECCEC} \arrow[d] \\
\text{AECAECCEC} \arrow[d] \\
\text{ACACCC} \arrow[d] \\
\text{EECC} \arrow[d] \\
\text{ECC} \arrow[d] \\
\text{CC}
\end{tikzcd}$$

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



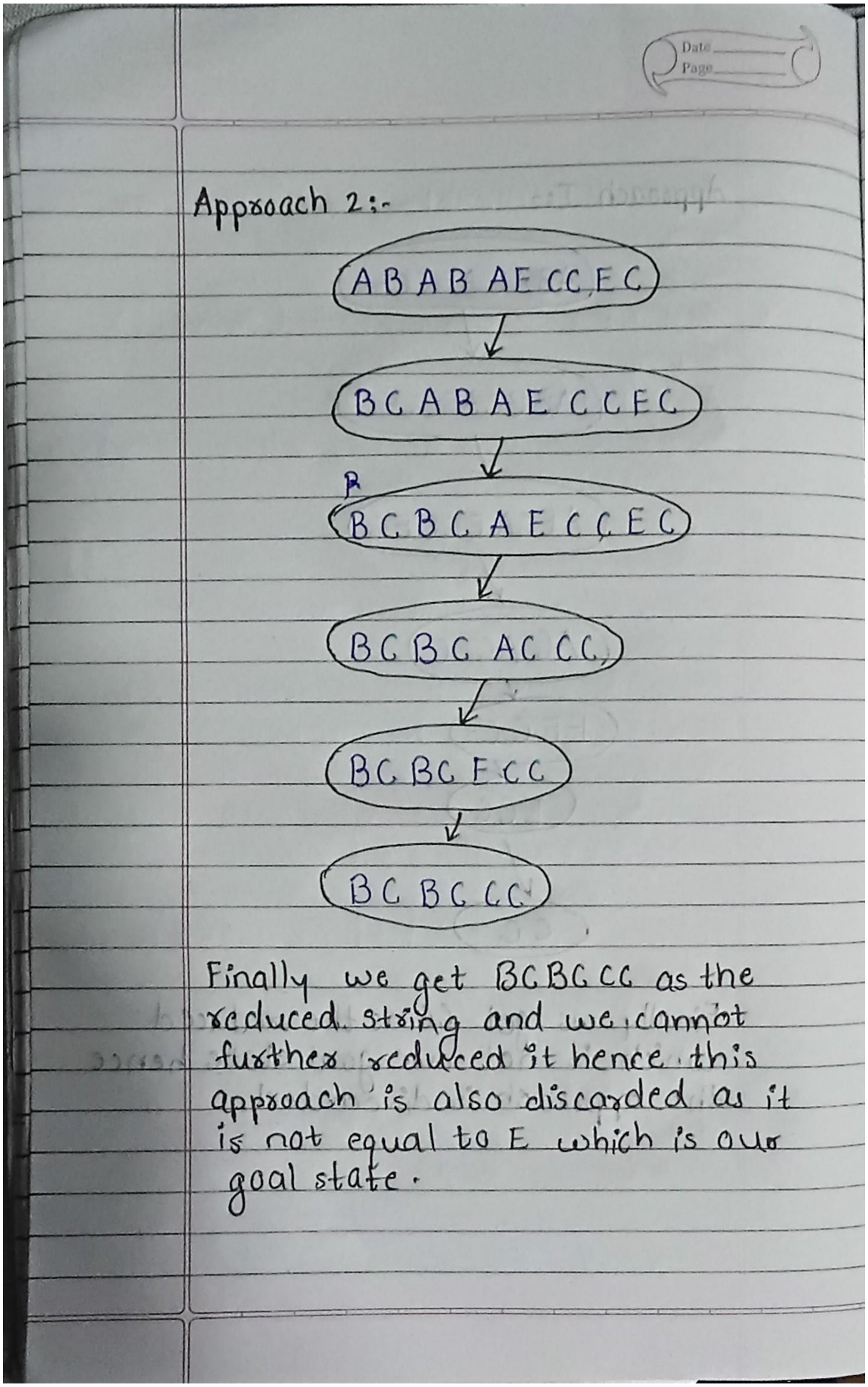
Step 5:- Converting EC to C
ECC
Step 6:- Converting EC to C
CC
So we can conclude as this is the least possible reduced by this method so we can understand that this method doesn't give solution so we need to try an alternative method.

Approach 2:-
ABABAECCEC
Step 1:- Converting AB to BC
BCABAECCEC
Step 2:- Converting AB to BC
BCBCAECCEC

Step 3:- Converting EC to C.
BCBCACCC
~~We cannot~~
Step 4:- Converting AC to E.
BCBCECC
Step 5:- Converting EC to C

BCBCCC
We cannot further reduce the string

Vyankatesh Sanjay Bura
2020BTECS00035



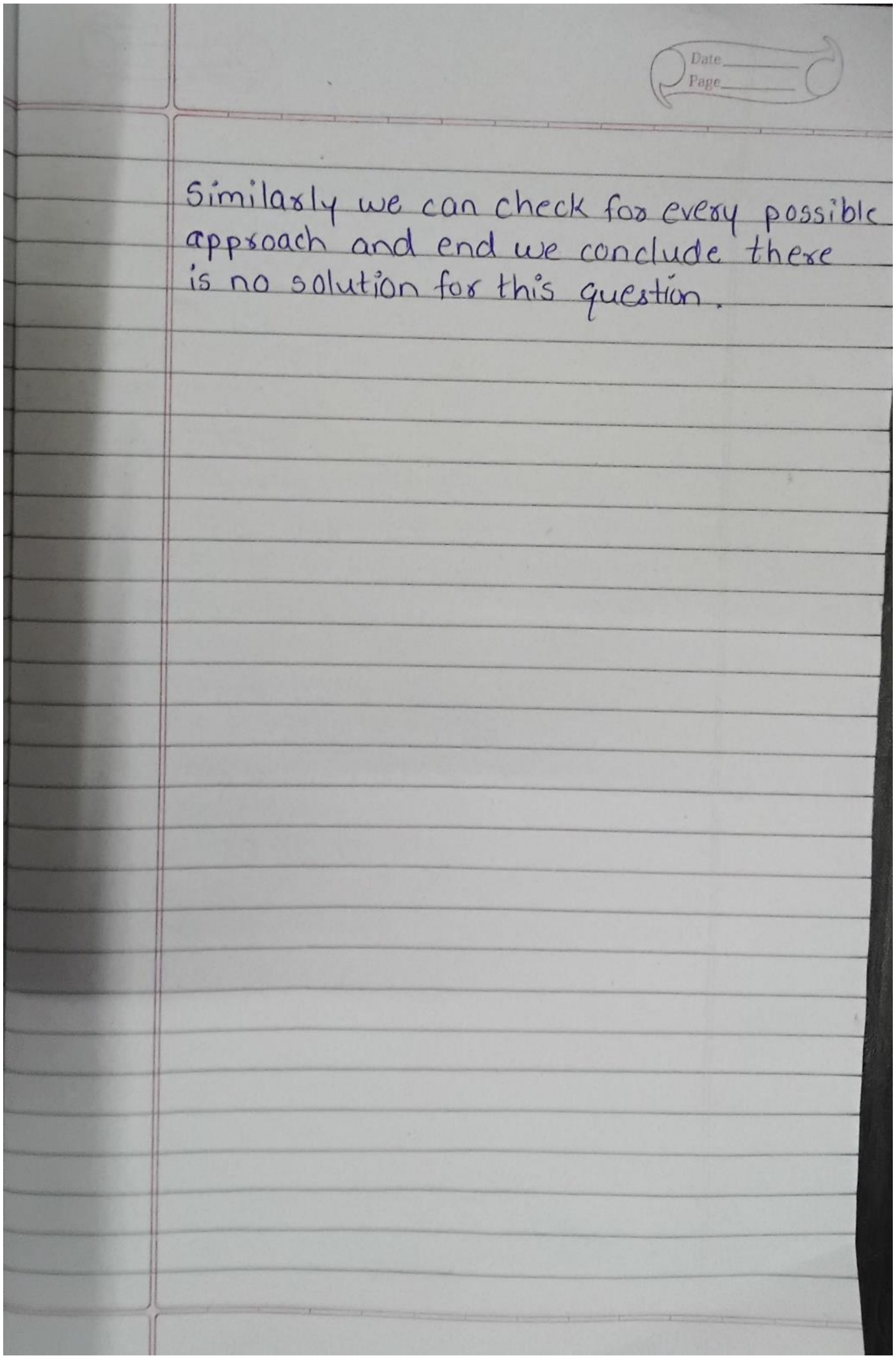
Date______
Page______

Approach 2:-

$$\begin{tikzcd}
\boxed{ABAB\,AE\,CC\,EC} \arrow[d] \\
\boxed{BC\,AB\,AE\,CCEC} \arrow[d] \\
\boxed{BC\,BC\,AE\,CCEC} \arrow[d] \\
\boxed{BC\,BC\,AC\,CC} \arrow[d] \\
\boxed{BC\,BC\,E\,CC} \arrow[d] \\
\boxed{BC\,BC\,CC}
\end{tikzcd}$$

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

Vyankatesh Sanjay Bura
2020BTECS00035

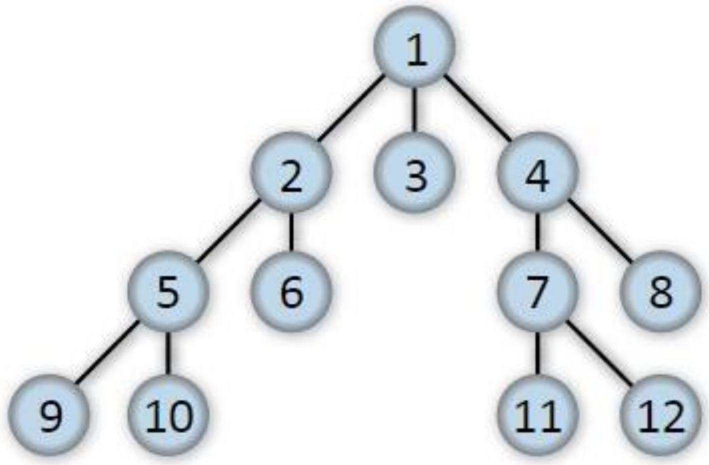


Date

Page

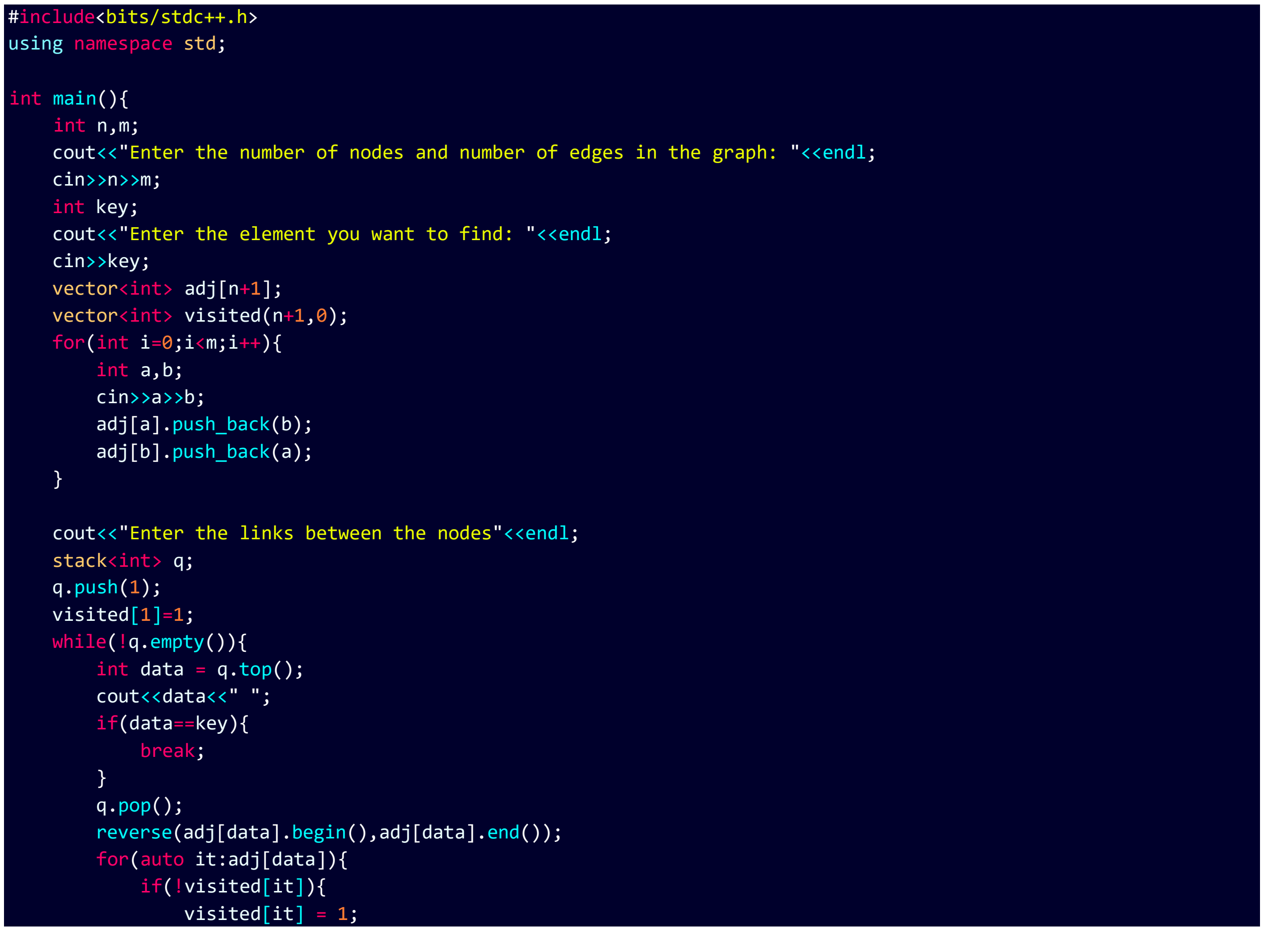
Similarly we can check for every possible approach and end we conclude there is no solution for this question.

3. For the tree given below:

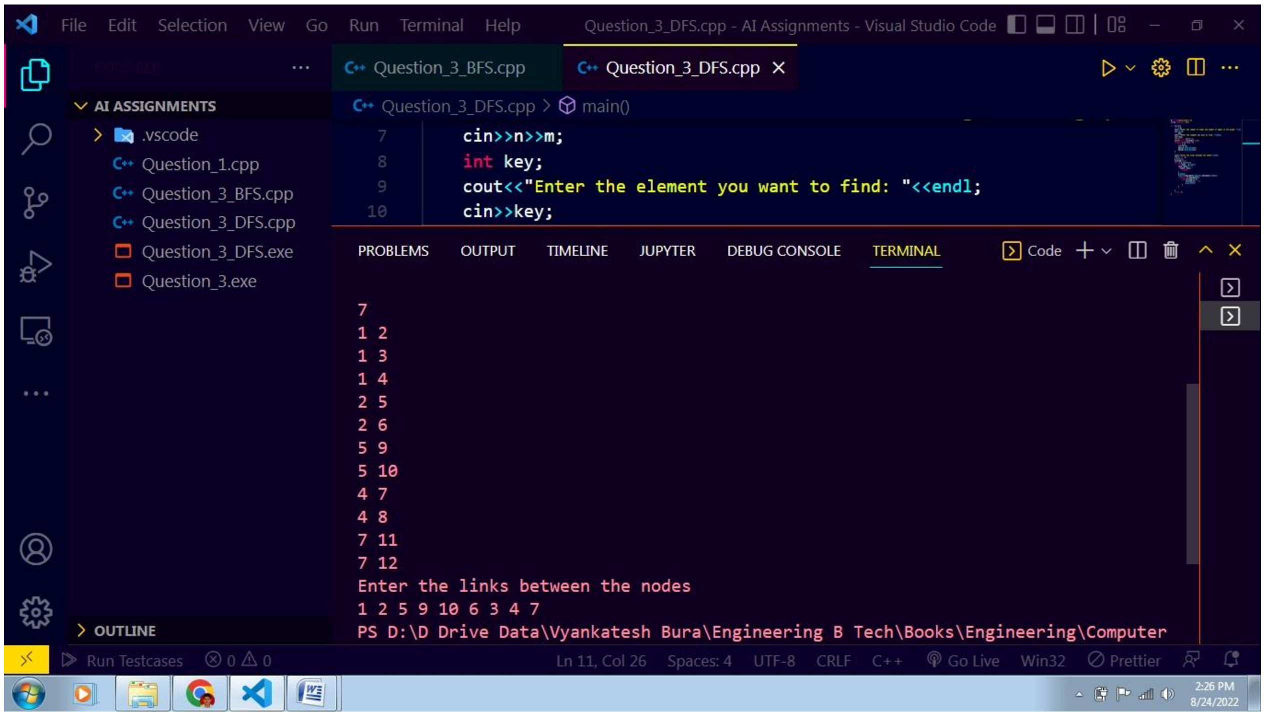


If starting state is 1 and goal state is 7, implement BFS and DFS search strategies for above tree. Which searching strategy will be best for this problem? Justify your answer with proper explanation.

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);
            }
        }

    }
    return 0;
}
```

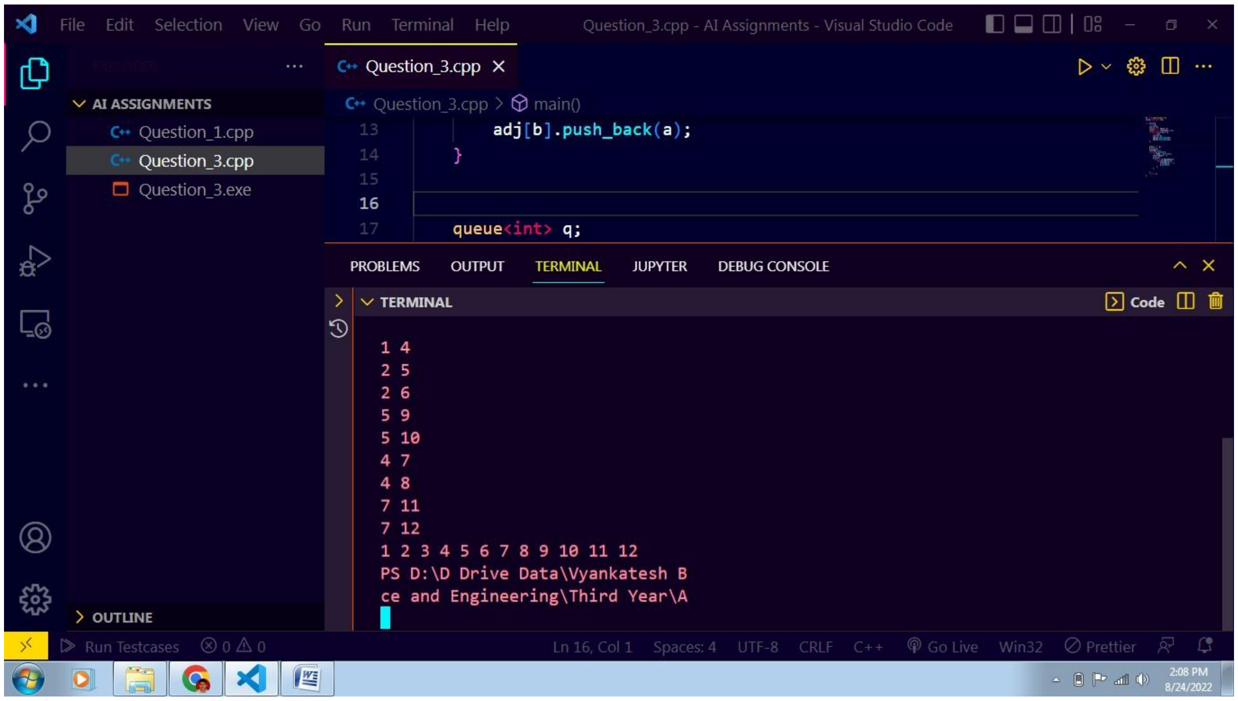


```
        cin>>n>>m;
        int key;
        cout<<"Enter the element you want to find: "<<endl;
        cin>>key;
```

```
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
PS D:\D Drive Data\Vyankatesh Bura\Engineering B Tech\Books\Engineering\Computer
```

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 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;
}
```

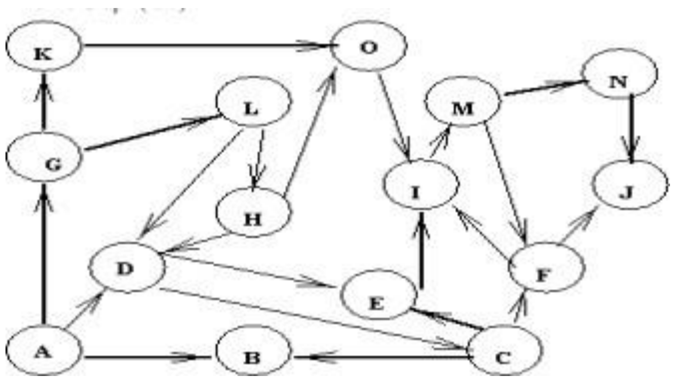

File Edit Selection View Go Run Terminal Help
Question_3.cpp - AI Assignments - Visual Studio Code
AI ASSIGNMENTS
Question_1.cpp
Question_3.cpp
Question_3.exe
OUTLINE
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
Ln 16, Col 1 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier

```
        adj[b].push_back(a);
    }

    queue<int> q;
```

```
1 4
2 5
2 6
5 9
5 10
4 7
4 8
7 11
7 12
1 2 3 4 5 6 7 8 9 10 11 12
PS D:\D Drive Data\Vyankatesh B
ce and Engineering\Third Year\A
```

Vyankatesh Sanjay Bura
2020BTECS00035

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



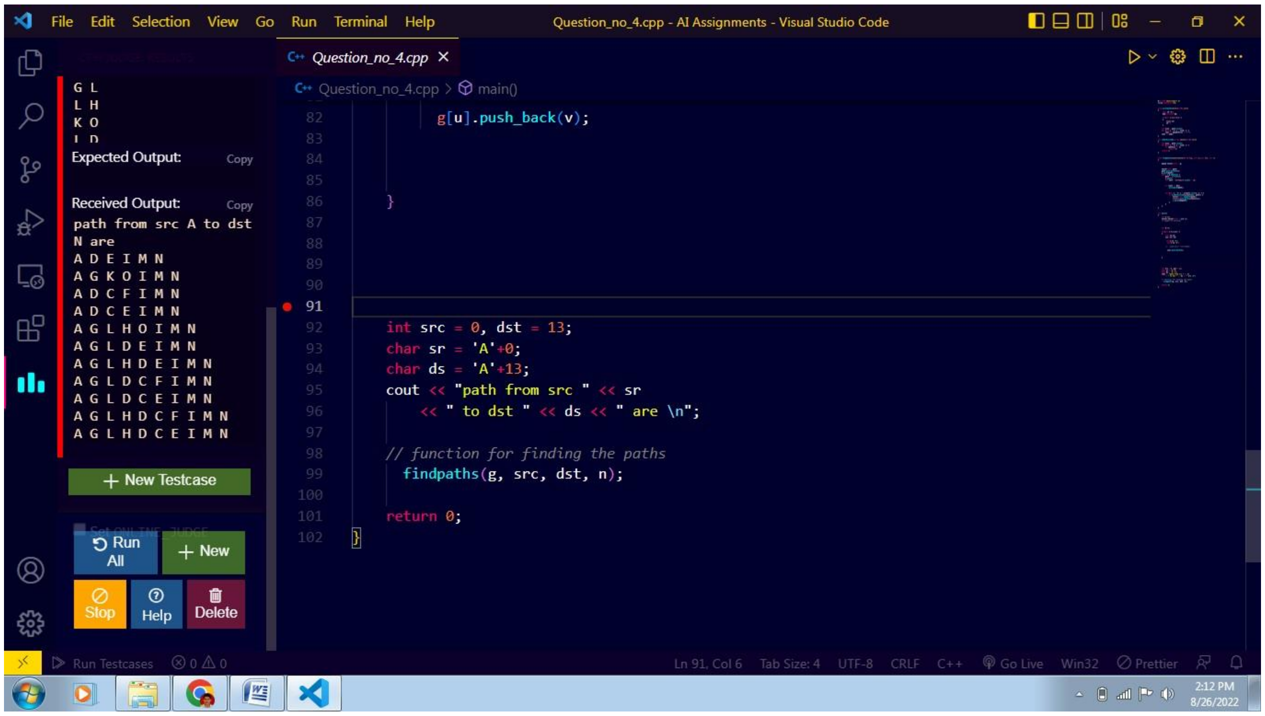
Identify different paths using

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;
}
```

Vyankatesh Sanjay Bura
2020BTECS00035



```
G L
L H
K O
I D
```

Expected Output:

Received Output:

```
path from src A to dst
N are
A D E I M N
A G K O I M N
A D C F I M N
A D C E I M N
A G L H O I M N
A G L D E I M N
A G L H D E I M N
A G L D C F I M N
A G L D C E I M N
A G L H D C F I M N
A G L H D C E I M N
```

```cpp
            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;
}
```

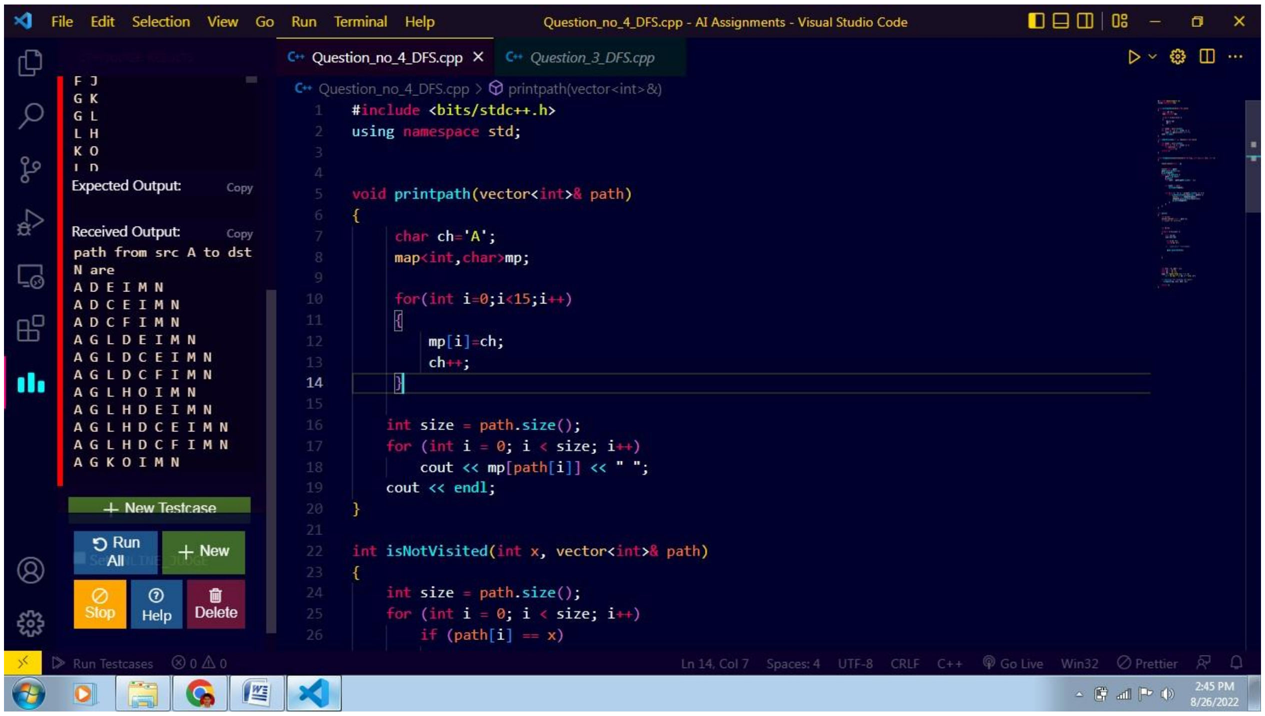
b. DFS with start state A and goal state N

Cpp Implementation:-

```cpp
#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";

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

    return 0;
}
```



Question_no_4_DFS.cpp - AI Assignments - Visual Studio Code

```
F J
G K
G L
L H
K O
I D
```

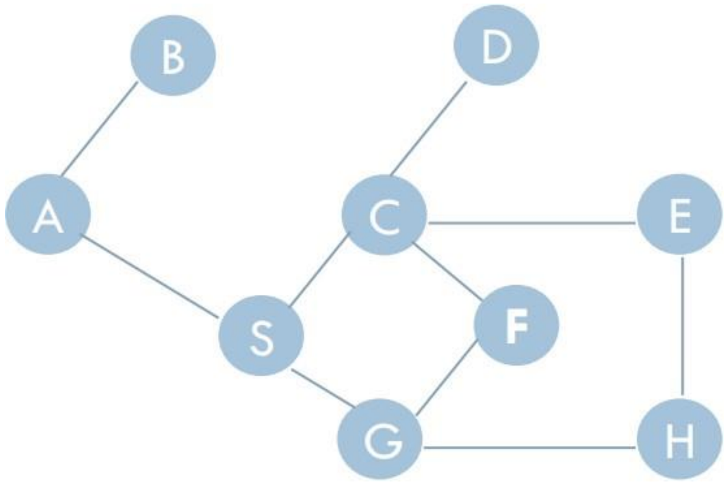
Expected Output:

Received Output:

```
path from src A to dst
N are
A D E I M N
A D C E I M N
A D C F I M N
A G L D E I M N
A G L D C E I M N
A G L D C F I M N
A G L H O I M N
A G L H D E I M N
A G L H D C E I M N
A G L H D C F I M N
A G K O I M N
```

```cpp
#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)
```

5. For the graph given below



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

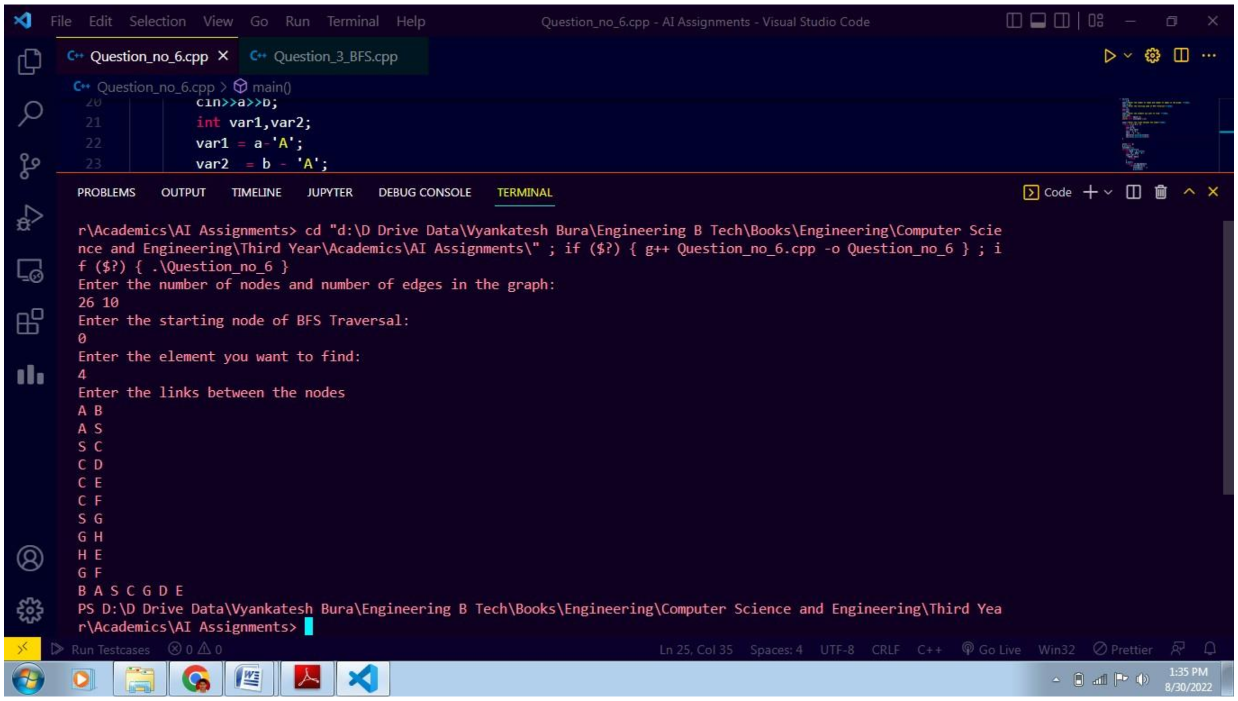
Answer:

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:


File Edit Selection View Go Run Terminal Help
Question_no_6.cpp - AI Assignments - Visual Studio Code
Question_no_6.cpp
Question_3_BFS.cpp
Question_no_6.cpp > main()
PROBLEMS OUTPUT TIMELINE JUPYTER DEBUG CONSOLE TERMINAL
Code
Run Testcases
Ln 25, Col 35 Spaces: 4 UTF-8 CRLF C++ Go Live Win32 Prettier
1:35 PM
8/30/2022

```
            cin>>a>>b;
            int var1,var2;
            var1 = a-'A';
            var2  = b - 'A';
```

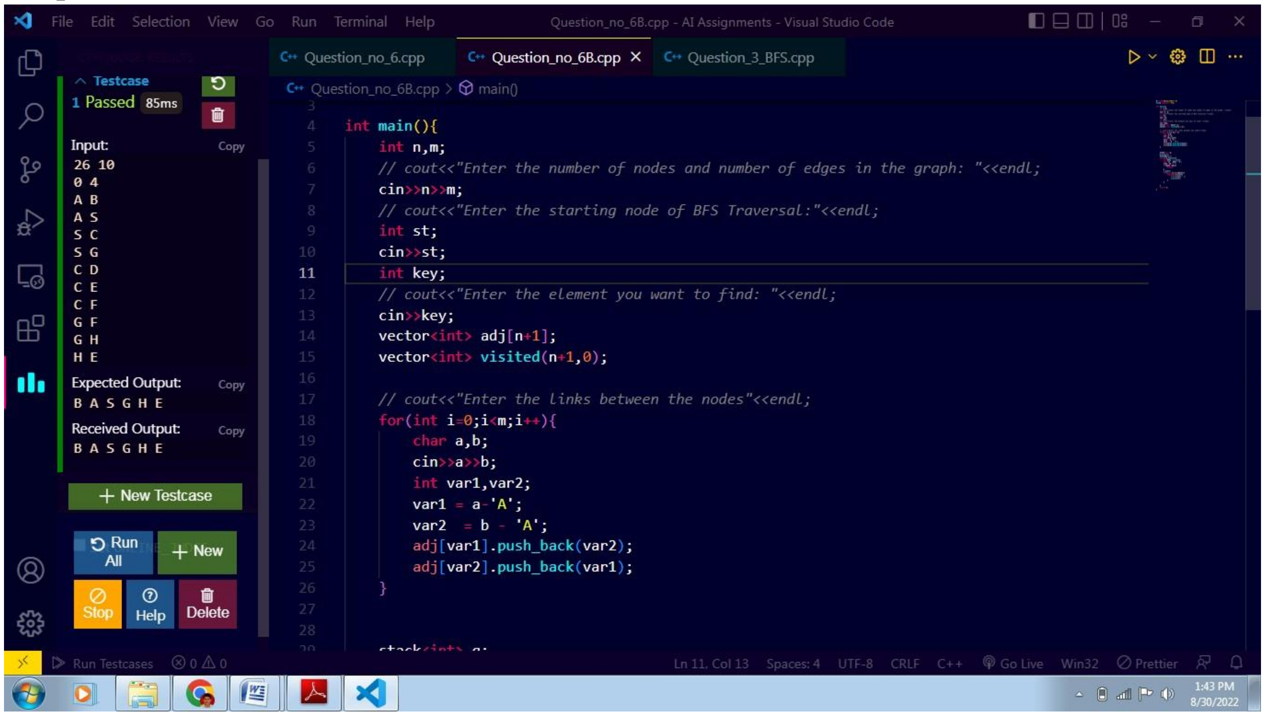
```
r\Academics\AI Assignments> cd "d:\D Drive Data\Vyankatesh Bura\Engineering B Tech\Books\Engineering\Computer Scie
nce and Engineering\Third Year\Academics\AI Assignments\" ; if ($?) { g++ Question_no_6.cpp -o Question_no_6 } ; i
f ($?) { .\Question_no_6 }
Enter the number of nodes and number of edges in the graph:
26 10
Enter the starting node of BFS Traversal:
0
Enter the element you want to find:
4
Enter the links between the nodes
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
PS D:\D Drive Data\Vyankatesh Bura\Engineering B Tech\Books\Engineering\Computer Science and Engineering\Third Yea
r\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:



```
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;
```

1 Passed 85ms

Input:

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

Expected Output:

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

Received Output:

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

.---.