ASSIGNMENT NO. 7

TITLE: Represent a given graph using adjacency list /matrix to perform DFS and using adjacency list to perform BFS. Use the map of the area around the college as the graph. Identify the prominent landmarks as nodes and perform DFS and BFS on that.

PROGRAM:

```
#include<iostream>
#include<string.h>
#include<stdlib.h>
using namespace std;
struct vertex
{
        char name[15];
        int visit;
        struct vertex *next;
};

struct vertex * lmark[20];

struct vertex *newnode(char nm[15])
{   struct vertex *p;

        p=(struct vertex*)malloc(sizeof(struct vertex));
        strcpy(p->name,nm);
        p->visit=0;
        p->next=NULL;

        return(p);
}
int search(char x[15],int n){
        int i;
        for(i=0;i<n;i++) {
                if(strcmp(lmark[i]->name,x)==0)
                return i;
        }}
void DFS(char s[15],int n) {
struct vertex *temp;
char stack[10][15],x[15];
int top=-1,i;
cout<<endl<<"DFS of given graph is"<<endl;
top++;
strcpy(stack[top],s); //pushing
        while(top!=-1) {
                strcpy(x,stack[top]);//poping
                top--;
                i=search(x,n);
                if(lmark[i]->visit==0) {
                        cout <<lmark[i]->name<<endl;
                        lmark[i]->visit=1;
                }
                else
                continue;
                temp=lmark[i]->next;
           while(temp!=NULL) {
                    i=search(temp->name,n);
                        if(lmark[i]->visit==0) {
                                top++;
                                strcpy(stack[top],temp->name);
                    }
                        temp=temp->next;
                }}}
void BFS(char s[15],int n) {
struct vertex *temp;
char Queue[10][15],x[15];
int front=-1,rear=-1,i;
cout<<endl<<"BFS of given graph is"<<endl;
if(rear==-1)
front++;
rear++;
strcpy(Queue[rear],s);//insert in queue
        while(rear!=-1) {
                strcpy(x,Queue[front]);//delete from queue
                front++;
                if(front>rear)
                front=rear=-1;
                i=search(x,n);
                if(lmark[i]->visit==0) {
                        cout <<lmark[i]->name<<endl;
                        lmark[i]->visit=1;
                } else
                continue;
                temp=lmark[i]->next;
           while(temp!=NULL) {
                        i=search(temp->name,n);
                        if(lmark[i]->visit==0)
                        {
                                if(rear==0)
                                front=0;
                                rear++;
                                strcpy(Queue[rear],temp->name);

                        }
                        temp=temp->next;
                }}}

int main()
{
int n,e,i,i1,i2,choice;
char v1[15],v2[15],s[15],x[15],nm[15];
struct vertex *p,*q,*temp,*m;
cout<<"Enter number of vertices"<<endl;
cin>>n;
for(i=0;i<n;i++)
{       cout<<"Enter name of vertex" <<i+1<<endl;
        cin>>nm;
        p=newnode(nm);
```

```
   lmark[i]=p;
            }
cout<<"Enter number of edges"<<endl;
cin>>e;
for(i=0;i<e;i++)
{
            cout<<"Enter edge"<<i+1<<endl;
            cin>>v1>>v2;
            i1=search(v1,n);
            i2=search(v2,n);
            p=newnode(v2);
            q=newnode(v1);
            temp=lmark[i1];
            while(temp->next!=NULL)
            {
                        temp=temp->next;
            }
            temp->next=p;
            temp=lmark[i2];
            while(temp->next!=NULL)
            {
                        temp=temp->next;
            }
            temp->next=q;
} cout<<endl;
cout<<"Adjacency list is"<<endl;
 for(i=0;i<n;i++)
 {
          temp=lmark[i];
            while(temp!=NULL)
            {   cout<<temp->name<<"->";
                        temp=temp->next;
            }
            cout<<endl;
}
cout<<endl<<"Enter starting node"<<endl;
cin>>s;
cout<<endl;
while(1) {
            cout<<"\nMenu:\n 1.DFS\n2.BFS\n3.Exit\n";
            cout<<"Enter your choice\n";
            cin>>choice;
            switch(choice)
            { case 1:
                        for(i=0;i<n;i++)
                        {
                         lmark[i]->visit=0; }
                        DFS(s,n);
                        break;
                        case 2:
                        for(i=0;i<n;i++) {
                         lmark[i]->visit=0;
            }
                        BFS(s,n);
```

```
                    break;
                    case 3:
                    exit(0);
          }//switch }//while }
```

OUTPUT:

```
Enter number of vertices
5
Enter name of vertex1
1
Enter name of vertex2
2
Enter name of vertex3
3
Enter name of vertex4
4
Enter name of vertex5
5
Enter number of edges
4
Enter edge1
1 2
Enter edge2
2 3
Enter edge3
3 4
Enter edge4
4 5
Adjacency list is
1->2->
2->1->3->
3->2->4->
4->3->5->
5->4->
Enter starting node
4
Menu:
 1.DFS
2.BFS
3.Exit
Enter your choice
1

DFS of given graph is
4
5
3
2
1
```