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

Program:

```
// Adjacency List
#include<iostream>
using namespace std;
#define MAX 10
#define TRUE 1
#define FALSE 0
class lgra {
        private:
                struct node1 {
                        int vertex;
                        struct node1 *next;
                };
                node1 *head[MAX];
                int visited[MAX];
        public:
        lgra();
        void create();
        void dfs(int);
};
lgra::lgra() {
        int v1;
        for(v1=0;v1<MAX;v1++)
                visited[v1]=FALSE;
                for(v1=0;v1<MAX;v1++)
                        head[v1]=NULL;
}
void lgra::create() {
        int v1,v2;
        char ans;
        node1 *N,*first;
        cout<<"Enter the vertices no. beginning with 0";
        do{
                cout<<"\nEnter the Edge of a graph\n";
                cin>>v1>>v2;
                if(v1>=MAX || v2>=MAX)
                        cout<<"Invalid Vertex Value\n";
                else {
                        N = new node1;
                        if (N==NULL)
                                cout<<"Insufficient Memory\n";
```

```
N->vertex=v2;
N->next=NULL;
first=head[v1];
if (first==NULL)
        head[v1]=N;
else{
        while(first->next!=NULL)
                first=first->next;
                first->next=N;
                }
                N=new node1;
                if (N==NULL)
                        cout<<"Insufficient Memory\n";
                        N->vertex=v1;
                        N->next=NULL;
                        first=head[v2];
                        if (first==NULL)
                                head[v2]=N;
                        else {
                                while(first->next!=NULL)
                                first=first->next;
                                first->next=N;
                        }
        }
                cout<<"\n Want to add more edges?(y/n)";
                cin>>ans;
        }while(ans=='y');
}
void lgra::dfs(int v1) {
        node1 *first;
        cout<<endl<<v1;
        visited[v1]=TRUE;
        first=head[v1];
        while(first!=NULL)
                if (visited[first->vertex]==FALSE)
                        dfs(first->vertex);
                else
                        first=first->next;
}
int main() {
        int v1;
        lgra g;
        g.create();
        cout<<endl<<"Enter the vertex from where you want to traverse:";
        cin>>v1;
        if(v1>=MAX)
                cout<<"Invalid Vertex\n";
```

```
else {
        cout<<"The Dfs of the graph:";
        g.dfs(v1);
}}
```