Practical No: - 12 Write a program to implement breadth first and depth first traversal.

A. Breadth first traversal

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;
int a[10][10],n,j,i,status[10],rear,front;
char node[10],q[10],start;
class BFS
{
public:
void find_bfs();
void get();
void put();
};
void BFS::find_bfs()
{
cout<<"\n Enter the no. of nodes :-";
cin>>n;
cout<<"\n Enter the nodes :-";
for(i=0;i<n;i++)
{
cin>>node[i];
status[i]=1;
}
cout<<"\n Enter the adjacency matrix :-";
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
cin>>a[i][j];
}
}
}
void BFS::get()
{
cout<<"\n Enter the starting node :-";
cin>>start;
```

```
rear=0,front=0;
for(i=0;i<n;i++)
{
q[i]=' ';
}
for(i=0;i<n;i++)
{
if(node[i]==start)
{
status[i]=2;
q[front]=start;
break;
}
}
}
void BFS::put()
{
cout<<"\n BFS Traversal is :-";
while(q[front]!=' '&&front<n)
{
for(i=0;i<n;i++)
{
if (node[i]==q[front])
{
break;
}
}
for(j=0;j<n;j++)
{
if(a[i][j]==1&&status[j]==1)
{
rear=rear+1;
q[rear]=node[j];
status[j]=2;
}
}
cout<<endl<<q[front];
for(j=0;j<n;j++)
{
if(node[i]==q[front])
```

```
{
status[i]=3;
break;
}
}
q[front]=' ';
front=front+1;
}
}
void main()
{
//clrscr();
BFS obj;
obj.find_bfs();
obj.get();
obj.put();
getch();
}
```

/*OUTPUT*/

```
 Enter the no. of nodes :-6
 Enter the nodes :-A B C D E F

 Enter the adjacency matrix :-
0 1 1 0 0 0
0 0 0 1 1 0
0 0 0 0 0 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

 Enter the starting node :-A

 BFS Traversal is :-
A
B
C
D
E
F
```

B. Depth first traversal

```
#include<iostream>
#include<conio.h>
using namespace std;
#define MAX 10
class DFS
{
private:
int n;
int adj[MAX][MAX];
int visited[MAX];
public:
void dfs(int);
void readmatrix();
};
void DFS::readmatrix()
{
int i,j;
cout<<"\n Enter the number of vertices in the graph :- ";
cin>>n;
cout<<"\n Enter the Adjency Matrix \n\n ";
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
    cin>>adj[i][j];
for(i=1;i<=n;i++)
    visited[i]=0;
}
void DFS::dfs(int source)
{
int i; visited[source]=1; cout<<source<<" ";
for(i=1;i<=n;i++)
if(adj[source][i]&&!visited[i])
    dfs(i);
}
int main()
{
int source;
//clrscr();
DFS depth;
```

```
depth.readmatrix();
cout<<"\n Enter The Source : ";
cin>>source;
cout<<"\n The Visited in the DFS order is :- ";
depth.dfs(source);
getch();
return 0;
}
```

/*OUTPUT*/

```
 Enter the number of vertices in the graph :- 6
 Enter the Adjency Matrix

0 1 1 0 0 0
0 0 0 1 1 0
0 0 0 0 0 1
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

Enter The Source : 1
The Visited in the DFS order is :- 1 2 4 5 3 6
```