**Practical No: 6**

**Program**:

#include<iostream>

#include<list>

#include<queue>

using namespace std;

class Graph

{

int V;

list<int>\*adjList;

bool\*\*adjMatrix;

public:

Graph(int V)

{

this->V=V;

adjList=new list<int>[V];

adjMatrix=new bool\*[V];

for(int i=0;i<V;i++)

{

adjMatrix[i]=new bool[V];

for (int j=0;j<V; j++) {

adjMatrix[i][j]=false;

}

}

}

void addEdge(int v,int w)

{

adjList[v].push\_back(w);

adjList[w].push\_back(v);

adjMatrix[v][w]=true;

adjMatrix[w][v]=true;

}

void DFSUtilMatrix(int v,bool visited[])

{

visited[v]=true;

cout <<v<<" ";

for (int i=0;i<V;i++)

{

if (adjMatrix[v][i] && !visited[i])

{

DFSUtilMatrix(i,visited);

}

}

}

void DFSMatrix(int v)

{

bool\*visited=new bool[V];

for(int i=0;i<V;i++)

visited[i]=false;

DFSUtilMatrix(v,visited);

cout << endl;

}

void BFSList(int s)

{

bool\*visited=new bool[V];

for(int i=0;i<V;i++)

visited[i]=false;

queue<int>queue;

visited[s]=true;

queue.push(s);

while(!queue.empty())

{

s=queue.front();

cout<<s<<" ";

queue.pop();

for(list<int>::iterator i=adjList[s].begin();i!=adjList[s].end();++i)

{

if (!visited[\*i])

{

visited[\*i]=true;

queue.push(\*i);

}

}

}

cout<<endl;

}

};

int main()

{

Graph g(6);

g.addEdge(0,1);

g.addEdge(0,2);

g.addEdge(0,3);

g.addEdge(1,4);

g.addEdge(2,5);

g.addEdge(3,5);

cout<<"DFS (starting from vertex 0):\n";

g.DFSMatrix(0);

cout<<"BFS (starting from vertex 0):\n";

g.BFSList(0);

return 0;

}

*OUTPUT :*

DFS (starting from vertex 0):

0 1 4 2 5 3

BFS (starting from vertex 0):

0 1 2 3 4 5