**AI - LAB Experiment**

- ASHISH KUMAR  
 **-** 2K18/SE/041

**Ques : Write a program on breadth first search, depth first search (uninformed search techniques).**

**DFS (Depth First Search)**

CODE:

#include<bits/stdc++.h>

using namespace std;

class Graph

{

int V;

list<int> \*adj;

void DFSUtil(int v, bool visited[]);

public:

Graph(int V);

void addEdge(int v, int w);

void DFS(int v);

};

Graph::Graph(int V)

{

this->V = V;

adj = new list<int>[V];

}

void Graph::addEdge(int v, int w)

{

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

}

void Graph::DFSUtil(int v, bool visited[])

{

visited[v] = true;

cout << v << "->";

list<int>::iterator i;

for (i = adj[v].begin(); i != adj[v].end(); ++i)

if (!visited[\*i])

DFSUtil(\*i, visited);

}

void Graph::DFS(int v)

{ bool \*visited = new bool[V];

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

visited[i] = false;

DFSUtil(v, visited);

}

int main()

{

Graph g(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 3);

cout << "Following is Depth First Traversal"

" (starting from vertex 0 to destination 3) \n";

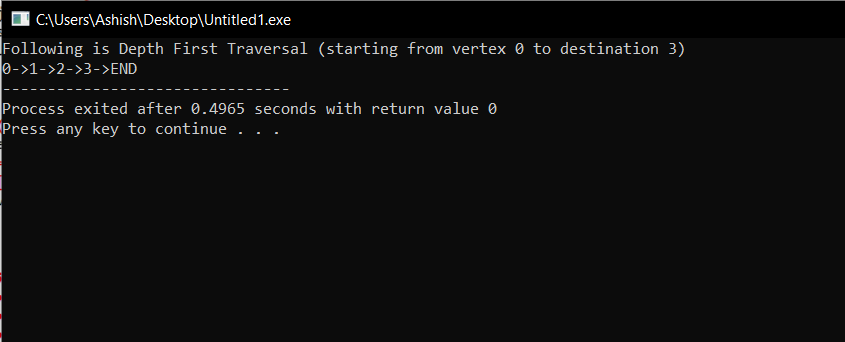
g.DFS(0);

cout<<"END";

return 0;

}

OUTPUT:



**BFS (Breadth First Search)**

CODE:

#include<bits/stdc++.h>

using namespace std;

class Graph

{

int V;

list<int> \*adj;

public:

Graph(int V);

void addEdge(int v, int w);

void BFS(int s);

};

Graph::Graph(int V)

{

this->V = V;

adj = new list<int>[V];

}

void Graph::addEdge(int v, int w)

{

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

}

void Graph::BFS(int s)

{

bool \*visited = new bool[V];

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

visited[i] = false;

list<int> queue;

visited[s] = true;

queue.push\_back(s);

list<int>::iterator i;

while(!queue.empty())

{

s = queue.front();

cout << s << "->";

queue.pop\_front();

for (i = adj[s].begin(); i != adj[s].end(); ++i)

{

if (!visited[\*i])

{

visited[\*i] = true;

queue.push\_back(\*i);

}

}

}

}

int main()

{

Graph g(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 3);

cout << "Following is Breadth First Traversal "

<< "(starting from vertex 2) \n";

g.BFS(2);

cout<<"END";

return 0;

}

OUTPUT:

