Date 4/9/20

Expt. No. _____ Page No. 19

Experiment -7

WAP to implement floyd warshall algorithm using dynamic programing.

```
#include <iostream>
using namespace std;
#define max 10
#define V 4
#define INF 999
int adj[max][max];
void printSolution (int adj[][V]);

Void warshall (int adj[][max]){
 int dist[V][V] = {0};
for ( int i=0; i<V; i++) {
   for (int j=0; j<V; j++) {
      dist [i][j] = adj [i][j]; }}

for ( int k=0; k<V; k++){
for (int i=0; i<V; i++) {
for (int j=0; j<V; j++) {
 if (dist [i] [k] + dist [k][j] < dis [i][j])
    dist [i][j] = dist [i][k] + dist [k][j]
 }}}
```

Teacher's Signature _____

```
printsolution (dist);
}
Void print solution (int dist[][V]){
Cout << " Following matrix shows the
shortest distances " between every pair of
vertices \n";
for( int i=0 ; i<V; i++) {
for( int j=0; j<V; j++) {
if ( dist[i][j] == 999)
  Cout << "INF";
else
  Cout << dist[i][j]; }
Cout << "\n";
}}

int main ( ){
int edges, i, j, origin, dest, max_edges;
int graph_type, weight
Cout << "\n enter 2 for undirected graph
and 1 for directed ";
Cin >> graph_type;
if (graph_type == 1)
{
  max_edges = max * (max-1)/2;
}
```

```
else
    max_edges = max * (max-1);
for ( i=1; i<=max_edges; i++) {
cout << "enter edge %d (-1 -1 to exit) :" i;
Cin >> origin;
Cin >> dest;
Cin >> weight;
if (origin == -1) && (dest == -1)
break;
if (origin >9 || dest >9 || origin<0 || dest<0)
    Cout << "\n invalid Vertex.";
    i--; }
else {
    adj [origin] [dest] = weight;
    if (graph_type == 2)
    adj [dest] [origin] = weight;
}
warshall (adj);
return 0;
}
```

Teacher's Signature

Output :-

```
enter 2 for undirected graph & 1 for directed : 1
enter edge 1 (-1 -1 to exit) : 0 1 3
enter edge 2 (-1 -1 to exit) : 0 0 0
enter edge 3 (-1 -1 to exit) : 0 2 999
enter edge 4 (-1 -1 to exit) : 0 3 6
enter edge 5 (-1 -1 to exit) : 1 0 3
enter edge 6 (-1 -1 to exit) : 1 1 0
enter edge 7 (-1 -1 to exit) : 1 2 2
enter edge 8 (-1 -1 to exit) : 1 3 999
enter edge 9 (-1 -1 to exit) : 2 0 5
enter edge 10 (-1 -1 to exit) : 2 1 999
enter edge 11 (-1 -1 to exit) : 2 3 0
enter edge 12 (-1 -1 to exit) : 3 0 1
enter edge 13 (-1 -1 to exit) : 3 1 2
enter edge 14 (-1 -1 to exit) : 3 2 999
enter edge 15 (-1 -1 to exit) : 3 3 0
enter edge 16 (-1 -1 to exit) : -1 -1 -1
```

Following matrix show the shortest distance between every pair of vertices

```
0  3  5  6
5  0  2  3
3  6  0  1
2  5  7  0
```