Name:- Yash Sanjay Kale
Reg No:- 2020BIT047

Practical :-8

1. Floyd Warshall Algorithm

```
#include <bits/stdc++.h>
using namespace std;
#define V 4
#define INF 99999
void printSolution(int dist[][V]);
void floydWarshall(int dist[][V])
{
int i, j, k;
for (k = 0; k < V; k++) {
for (i = 0; i < V; i++) {
for (j = 0; j < V; j++) {
if (dist[i][j] > (dist[i][k] + dist[k][j])
&& (dist[k][j] != INF
&& dist[i][k] != INF))
dist[i][j] = dist[i][k] + dist[k][j];
}
}
}
// Print the shortest distance matrix
printSolution(dist);
}
void printSolution(int dist[][V])
{
cout << "The 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] == INF)
cout << "INF"
<< " ";
else
cout << dist[i][j] << " ";
}
cout << endl;
}
}
int main()
{
int graph[V][V] = { { 0, 5, INF, 10 },
{ INF, 0, 3, INF },
{ INF, INF, 0, 1 },
{ INF, INF, INF, 0 } };
// Function call
floydWarshall(graph);
return 0;
}
```

2. Knapsack Algorithm

```
#include <bits/stdc++.h>
using namespace std;
```

```
// maximum of two integers
int max(int a, int b) { return (a > b) ? a : b; }
// Returns the max value than can put
int knapSack(int W, int wt[], int val[], int n)
{
if (n == 0 || W == 0)
return 0;
// in case weight of item is greater do not include it
if (wt[n - 1] > W)
return knapSack(W, wt, val, n - 1);
// return max of both
else
return max(
val[n - 1]
+ knapSack(W - wt[n - 1], wt, val, n - 1),
knapSack(W, wt, val, n - 1));
}
int main()
{
int profit[] = { 60, 100, 120 };
int weight[] = { 10, 20, 30 };
int W = 50;
int n = sizeof(profit) / sizeof(profit[0]);
cout << knapSack(W, weight, profit, n);
return 0;
}
```