FLAT ASSIGNMENT

NAME: KOTLA CHETAN KUMAR REDDY ROLL.NO : 21071A6731

TRAVELLING SALESPERSON PROBLEM :-

Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. Note the difference between Hamiltonian Cycle and TSP. The Hamiltonian cycle problem is to find if there exists a tour that visits every city exactly once. Here we know that Hamiltonian Tour exists (because the graph is complete) and in fact, many such tours exist, the problem is to find a minimum weight Hamiltonian Cycle.

ALGORITHM:

```
  Algorithm: Traveling-Salesperson-Problem
{
  Cost (1, {}, 1) = 0

   for s = 2 to n do
      for all subsets S belongs to {1, 2, 3, … , n} of size s
         Cost (s, S, 1) = Infinity

  for all i ∈ S and i ≠ 1
 Cost (i, S, j) = min {Cost (i, S – {i}, j) + dist(i, j) for j ∈ S and i ≠ j}

  Return min(i) Cost (i, {1, 2, 3, …, n}, j) + d(j, i)
}
```

CODE:

```
#include <bits/stdc++.h>

using namespace std;
#define V 4
#define MAX  1000000

int tsp(int graph[][V], int s)
{
    vector<int> vertex;
    for (int i = 0; i < V; i++)
        if (i != s)
            vertex.push_back(i);
    int min_cost = MAX;
    while(next_permutation(vertex.begin(), vertex.end()))
```

```
    {
        int current_cost = 0;
        int j = s;
        for (int i = 0; i < vertex.size(); i++)
        {
            current_cost += graph[j][vertex[i]];
            j = vertex[i];
        }
        current_cost += graph[j][s];
        min_cost = min(min_cost, current_cost);
        return min_cost;
    }
}

int main()
{
    int graph[][V] = { { 0, 10, 15, 20 },
                       { 5,  0,  9, 10 },
                       { 6, 13,  0, 12 },
                       { 8,  8,  9,  0 } };
    int s = 0;
    cout << "The minimum cost is "<<tsp(graph, s) << endl;
    return 0;
}
```

OUTPUT: