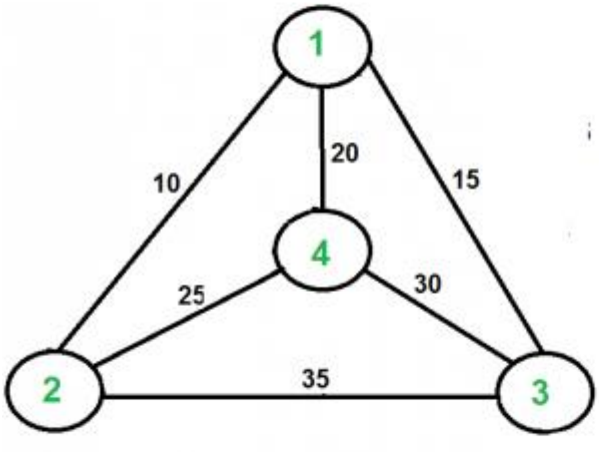
Experiment:-12 Traveling Salesman Problem (TSP) Implementation

Travelling Salesman Problem (TSP) : Given a set of cities and distances 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.
For example, consider the graph shown in the figure on the right side. A TSP tour in the graph is 1-2-4-3-1. The cost of the tour is 10+25+30+15 which is 80.
The problem is a famous NP-hard problem. There is no polynomial-time known solution for this problem

.



Examples:
Output of Given Graph:

minimum weight Hamiltonian Cycle :

10 + 25 + 30 + 15 := 80

CPP program to implement traveling salesman
// problem using naive approach.

```
#include <bits/stdc++.h>
using namespace std;
#define V 4

// implementation of traveling Salesman Problem
int travllingSalesmanProblem(int graph[][V], int s)
{
    // store all vertex apart from source vertex
    vector<int> vertex;
    for (int i = 0; i < V; i++)
        if (i != s)
            vertex.push_back(i);

    // store minimum weight Hamiltonian Cycle.
    int min_path = INT_MAX;
    do {

        // store current Path weight(cost)
        int current_pathweight = 0;

        // compute current path weight
        int k = s;
        for (int i = 0; i < vertex.size(); i++) {
            current_pathweight += graph[k][vertex[i]];
            k = vertex[i];
        }
        current_pathweight += graph[k][s];

        // update minimum
        min_path = min(min_path, current_pathweight);

    } while (
        next_permutation(vertex.begin(), vertex.end()));

    return min_path;
}
```

```
// Driver Code
int main()
{
    // matrix representation of graph
    int graph[][V] = { { 0, 10, 15, 20 },
                   { 10, 0, 35, 25 },
                   { 15, 35, 0, 30 },
                   { 20, 25, 30, 0 } };
    int s = 0;
    cout << travllingSalesmanProblem(graph, s) << endl;
    return 0;
}
```