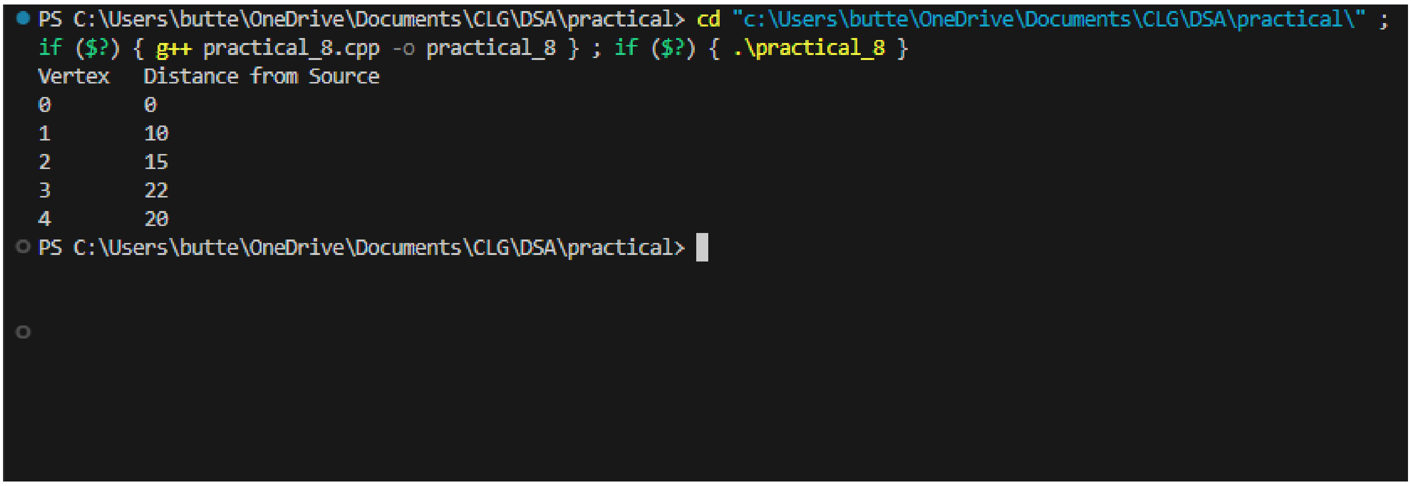
Practical 8
Source Code:-

```
/*A: City Hall
B: Library
C: Park
D: Museum
E: Restaurant

matrix
   A  B  C  D  E
A [  0,  10, 15, 0,  0]
B [  10, 0,  0,  12, 0]
C [  15, 0,  0,  0,  5]
D [  0,  12, 0,  0,  10]
E [  0,  0,  5,  10, 0]

*/
#include <iostream>
#include <vector>
#include <limits>
#include <utility>
using namespace std;

#define V 5 // Number of vertices

// Function to find the vertex with the minimum distance
int minDistance(const vector<int>& dist, const vector<bool>& sptSet) {
  int min = numeric_limits<int>::max(), min_index;
  for (int v = 0; v < V; v++) {
    if (!sptSet[v] && dist[v] <= min) {
      min = dist[v];
      min_index = v;
    }
  }
  return min_index;
}

// Function to implement Dijkstra's algorithm
void dijkstra(int graph[V][V], int src) {
  vector<int> dist(V, numeric_limits<int>::max());
  vector<bool> sptSet(V, false); // Shortest path tree set

  dist[src] = 0; // Distance from source to itself is 0

  for (int count = 0; count < V - 1; count++) {
    int u = minDistance(dist, sptSet);
    sptSet[u] = true; // Mark the picked vertex as processed

    // Update dist value of the neighboring vertices of the picked vertex
    for (int v = 0; v < V; v++) {
      if (!sptSet[v] && graph[u][v] && dist[u] != numeric_limits<int>::max() &&
        dist[u] + graph[u][v] < dist[v]) {
            dist[v] = dist[u] + graph[u][v];
         }
      }
   }

   // Print the shortest distances
   cout << "Vertex \t Distance from Source\n";
   for (int i = 0; i < V; i++) {
      cout << i << " \t " << dist[i] << endl;
   }
}

int main() {
   // Adjacency matrix representation of the graph
   int graph[V][V] = {
      {0, 10, 15, 0, 0},
      {10, 0, 0, 12, 0},
      {15, 0, 0, 0, 5},
      {0, 12, 0, 0, 10},
      {0, 0, 5, 10, 0}
   };

   // Run Dijkstra's algorithm from the source vertex A (0)
   dijkstra(graph, 0);

   return 0;
}
```

Output:-



```
PS C:\Users\butte\OneDrive\Documents\CLG\DSA\practical> cd "c:\Users\butte\OneDrive\Documents\CLG\DSA\practical\" ;
if ($?) { g++ practical_8.cpp -o practical_8 } ; if ($?) { .\practical_8 }
Vertex   Distance from Source
0        0
1        10
2        15
3        22
4        20
PS C:\Users\butte\OneDrive\Documents\CLG\DSA\practical>
```