#include <iostream>

using namespace std;

#define INF 999999

#define MAX\_VERTICES 100

class CityGraph

{

private:

int adjMatrix[MAX\_VERTICES][MAX\_VERTICES];

string landmarks[MAX\_VERTICES];

int numVertices;

int minDistance(int dist[], bool visited[])

{

int min = INF, minIndex = -1;

for (int v = 0; v < numVertices; v++)

{

if (!visited[v] && dist[v] < min)

{

min = dist[v];

minIndex = v;

}

}

return minIndex;

}

public:

CityGraph(int vertices)

{

numVertices = vertices;

for (int i = 0; i < vertices; i++)

{

for (int j = 0; j < vertices; j++)

{

adjMatrix[i][j] = INF;

}

adjMatrix[i][i] = 0;

}

}

void addLandmark(int index, string name)

{

if (index >= 0 && index < numVertices)

{

landmarks[index] = name;

}

}

void addEdge(int from, int to, int distance)

{

if (from >= 0 && from < numVertices && to >= 0 && to < numVertices)

{

adjMatrix[from][to] = distance;

adjMatrix[to][from] = distance;

}

}

void dijkstra(int source)

{

int dist[MAX\_VERTICES];

bool visited[MAX\_VERTICES];

int prev[MAX\_VERTICES];

for (int i = 0; i < numVertices; i++)

{

dist[i] = INF;

visited[i] = false;

prev[i] = -1;

}

dist[source] = 0;

for (int count = 0; count < numVertices - 1; count++)

{

int u = minDistance(dist, visited);

visited[u] = true;

for (int v = 0; v < numVertices; v++)

{

if (!visited[v] && adjMatrix[u][v] != INF &&

dist[u] + adjMatrix[u][v] < dist[v])

{

dist[v] = dist[u] + adjMatrix[u][v];

prev[v] = u;

}

}

}

cout << "\nShortest paths from " << landmarks[source] << ":\n";

for (int i = 0; i < numVertices; i++)

{

if (i != source)

{

cout << "To " << landmarks[i] << ": ";

if (dist[i] == INF)

{

cout << "No path exists\n";

}

else

{

cout << dist[i] << " units, Path: ";

int current = i;

string path = landmarks[i];

while (prev[current] != -1)

{

path = landmarks[prev[current]] + " -> " + path;

current = prev[current];

}

cout << path << "\n";

}

}

}

}

};

int main()

{

CityGraph graph(5);

graph.addLandmark(0, "Park");

graph.addLandmark(1, "Museum");

graph.addLandmark(2, "Library");

graph.addLandmark(3, "Restaurant");

graph.addLandmark(4, "Mall");

graph.addEdge(0, 1, 5);

graph.addEdge(0, 2, 4);

graph.addEdge(1, 2, 2);

graph.addEdge(1, 3, 3);

graph.addEdge(2, 3, 6);

graph.addEdge(2, 4, 8);

graph.addEdge(3, 4, 7);

graph.dijkstra(0);

return 0;

}