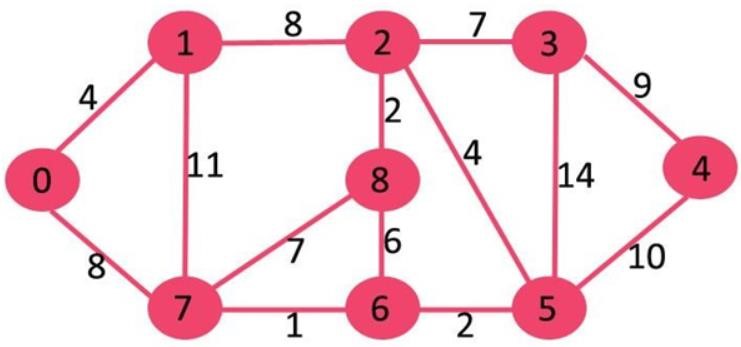
**Experiment - 5**

|  |  |
| --- | --- |
| **Student Name: Pankaj Singh Kanyal** | **UID:** 20BCS6668 |
| **Branch:** CSE AIML | **Section/Group:** AIML 4 B |
| **Semester:** 5th | **Date of Performance:** 11/10/2022 |
| **Subject Name:** Advanced Programming Lab | **Subject Code:** 20CSP-334 |

1. **Experiment Title/Problem Statement:**

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra’s algorithm.



1. **Requirements:**

IDE (Jetbrains CLion)

Laptop/PC

1. **Description:**

Dijkstra’s algorithm is very similar to Prim’s algorithm for minimum spanning tree. Like Prim’s MST, we generate a SPT (shortest path tree) with a given source as a root. We maintain two sets, one set contains vertices included in the shortest-path tree, other set includes vertices not yet included in the shortest-path tree. At every step of the algorithm, we find a vertex that is in the other set (set of not yet included) and has a minimum distance from the source.

1. **Algorithm** 
   1. Create a set sptSet (shortest path tree set) that keeps track of vertices included in the shortest-path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.
   2. Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.
   3. While sptSet doesn’t include all vertices

a) Pick a vertex u which is not there in sptSet and has a minimum distance value. b) Include u to sptSet.

c) Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if the sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

1. **Steps for experiment/practical:**

#include <iostream>

using namespace std;

#include <climits>

#define V 9

int minDistance(const int dist[], const bool sptSet[]) {

int min = 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;

}

void printSolution(int dist[]) {

cout << "Vertex \t Distance from Source" << endl; for (int i = 0; i < V; i++)

cout << i << " \t\t" << dist[i] << endl;

}

void dijkstra(int graph[V][V], int src) {

int dist[V]; bool sptSet[V]; for (int i = 0; i < V; i++)

dist[i] = INT\_MAX, sptSet[i] = false; dist[src] = 0; for (int count = 0; count < V - 1; count++) { int u = minDistance(dist, sptSet);

sptSet[u] = true; for (int v = 0; v < V; v++)

if (!sptSet[v] && graph[u][v] && dist[u] != INT\_MAX

&& dist[u] + graph[u][v] < dist[v]) dist[v] = dist[u] + graph[u][v];

}

printSolution(dist);

}

int main() { int graph[V][V] = {{0, 4, 0, 0, 0, 0, 0, 8, 0},

{4, 0, 8, 0, 0, 0, 0, 11, 0},

{0, 8, 0, 7, 0, 4, 0, 0, 2},

{0, 0, 7, 0, 9, 14, 0, 0, 0},

{0, 0, 0, 9, 0, 10, 0, 0, 0},

{0, 0, 4, 14, 10, 0, 2, 0, 0},

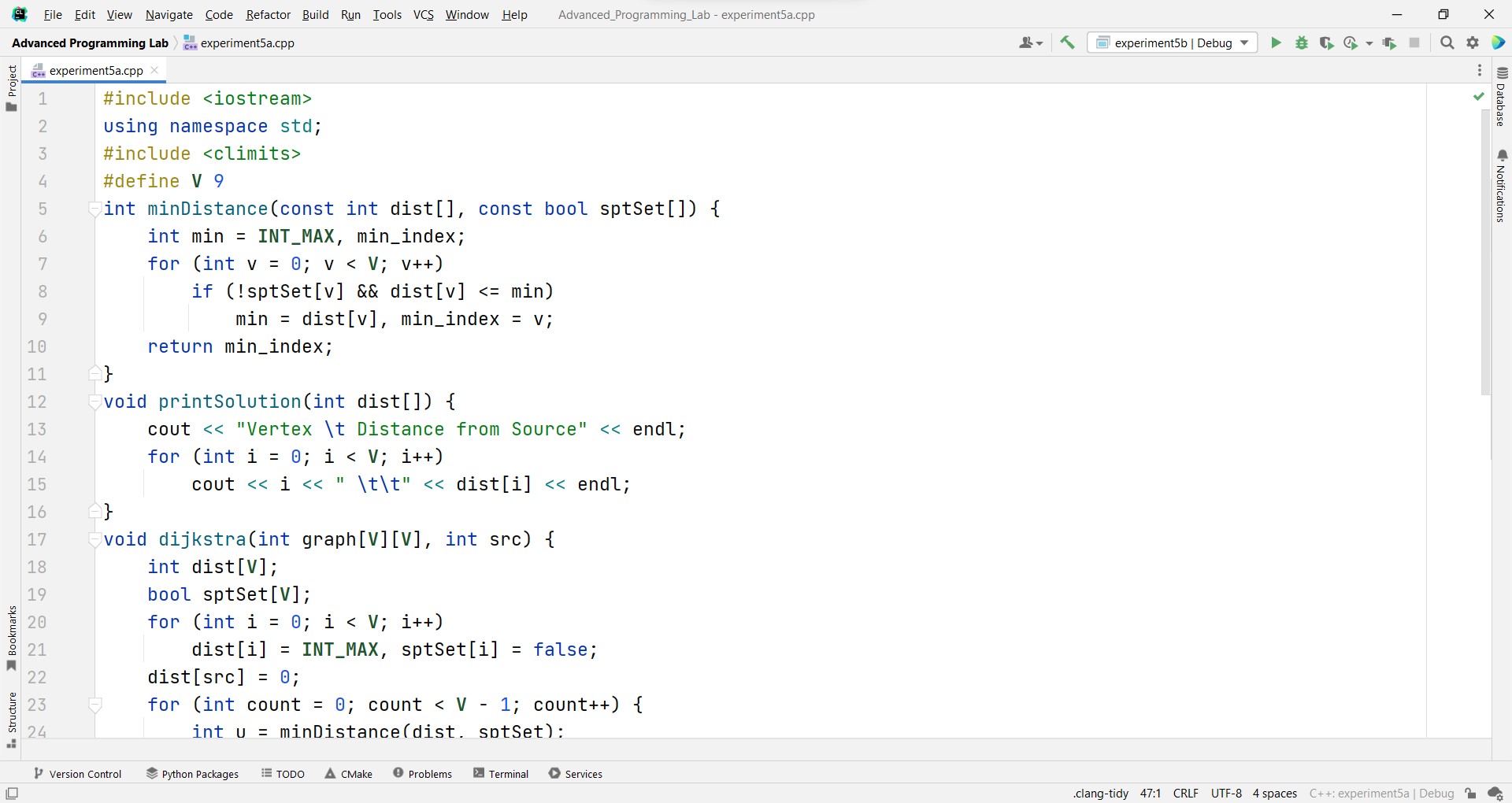
{0, 0, 0, 0, 0, 2, 0, 1, 6},

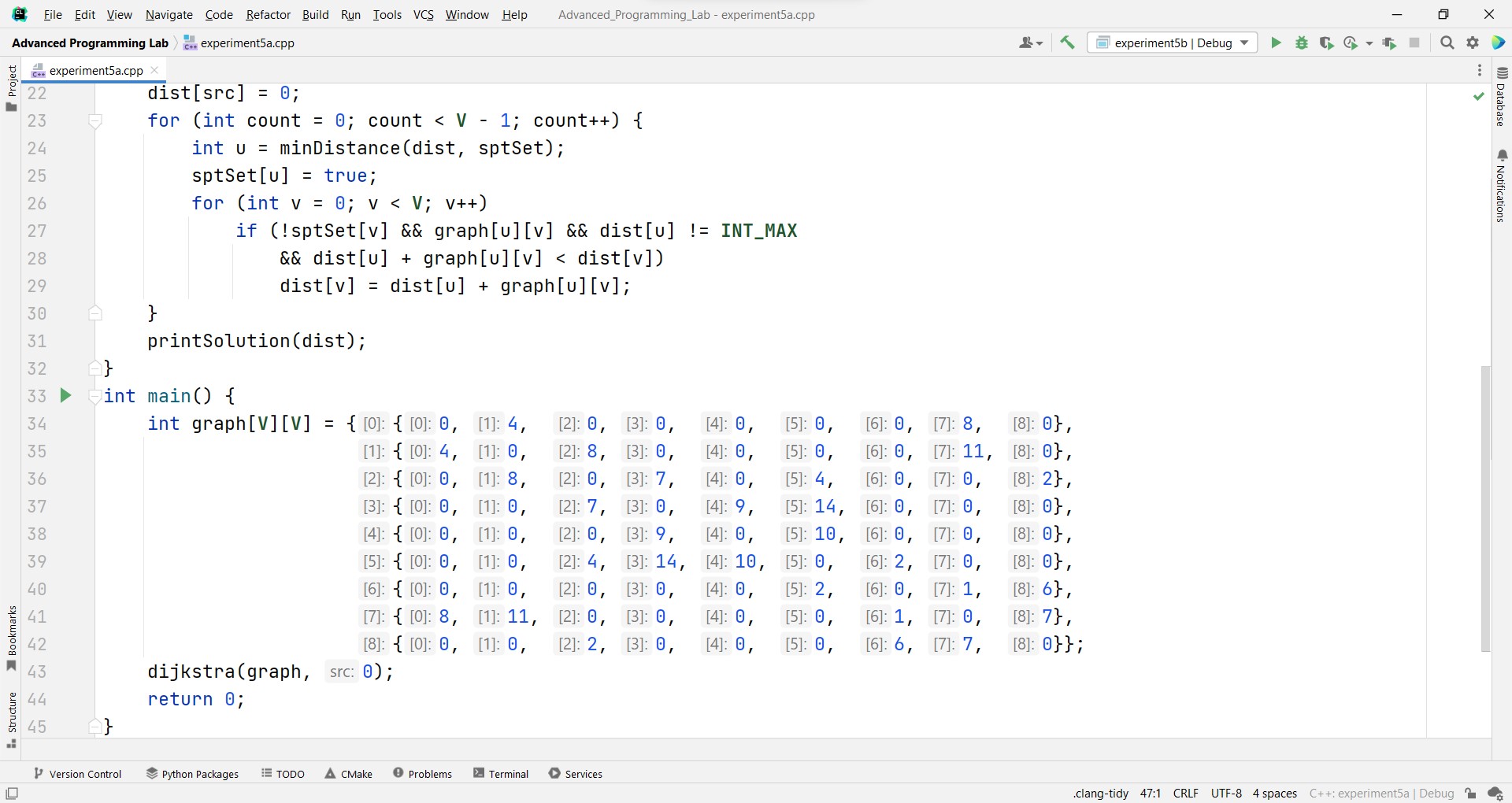
{8, 11, 0, 0, 0, 0, 1, 0, 7}, {0, 0, 2, 0, 0, 0, 6, 7, 0}};

dijkstra(graph, 0); return 0;

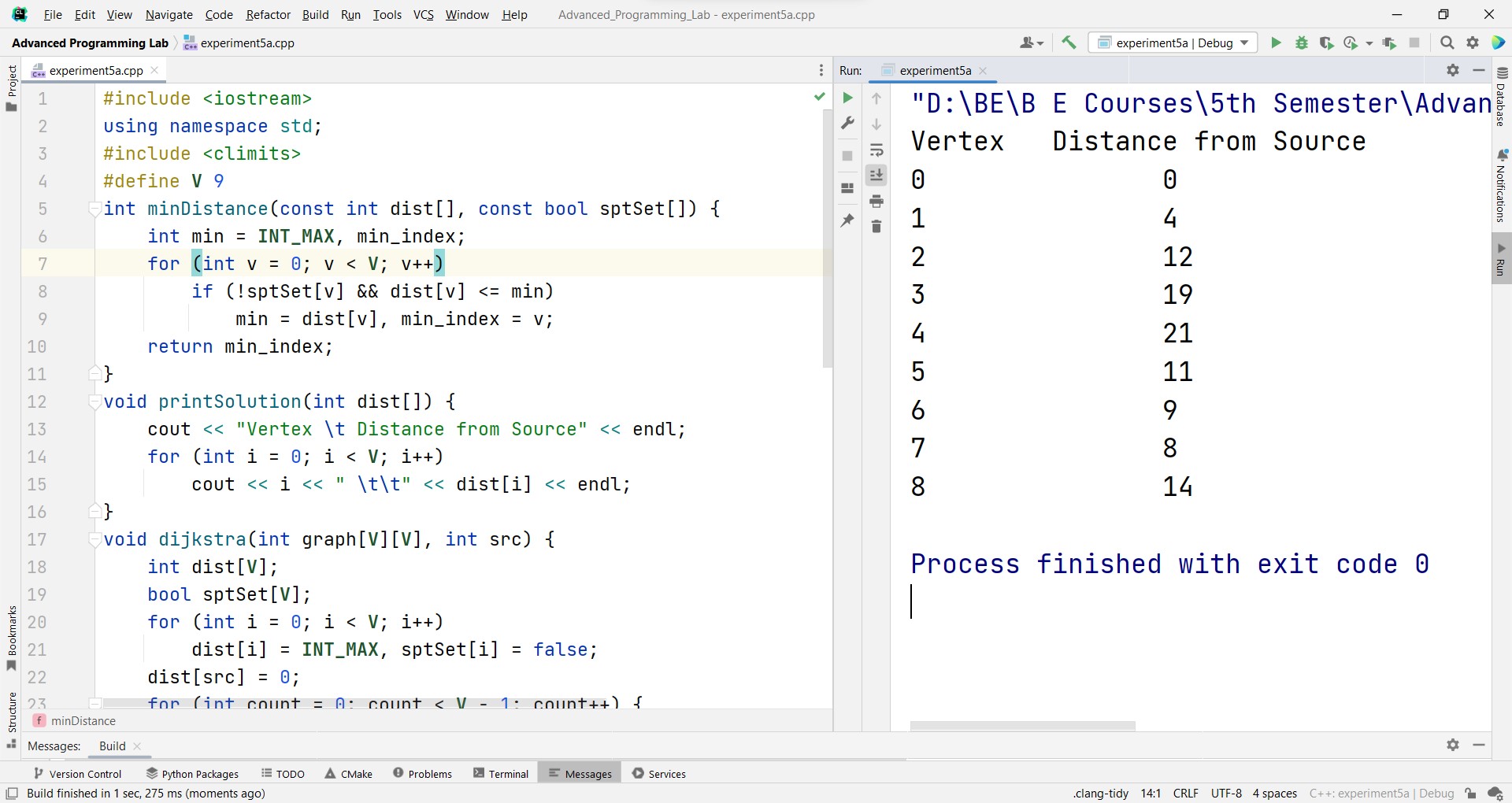
}

1. **Snapshots of Above steps**





1. **Result/Output/Writing Summary:**



1. **Learning Outcomes:** 
   1. Learned the concepts Dijkstra’s algorithm.
   2. Learned the concepts of Graphs.
   3. Learned to write a program for the the above problem.
   4. Learned to use Clion IDE.

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Parameters** | **Marks Obtained** | **Maximum Marks** |
| **1.** |  |  |  |
| **2.** |  |  |  |
| **3.** |  |  |  |