WEEK 14 - Implementation of Dijkstra's Algorithm

```
#include <stdio.h>
#include <stdbool.h>
#define MAX_VERTICES 10
#define INF 999999
int graph[MAX_VERTICES][MAX_VERTICES];
int vertices;
void createGraph() {
int i, j;
printf("Enter the number of vertices: ");
scanf("%d", &vertices);
printf("Enter the adjacency matrix:\n");
for (i = 0; i < vertices; i++) {
for (j = 0; j < vertices; j++) {
scanf("%d", &graph[i][j]);
}
}
}
int minDistance(int dist[], bool sptSet[]) {
int min = INF, min_index;
for (int v = 0; v < vertices; v++) {
if (sptSet[v] == false && dist[v] <= min) {
min = dist[v];
min_index = v;
}
}
return min_index;
}
void printSolution(int dist[]) {
printf("Vertex \t Distance from Source\n");
for (int i = 0; i < vertices; i++) {
```

```
printf("%d \t %d\n", i, dist[i]);
}
}void dijkstra(int src) {
int dist[vertices];
bool sptSet[vertices];
for (int i = 0; i < vertices; i++) {
dist[i] = INF;
sptSet[i] = false;
}
dist[src] = 0;
for (int count = 0; count < vertices - 1; count++) {
int u = minDistance(dist, sptSet);
sptSet[u] = true;
for (int v = 0; v < vertices; v++) {
if (!sptSet[v] && graph[u][v] && dist[u] != INF && dist[u] + graph[u][v] < dist[v])
{
dist[v] = dist[u] + graph[u][v];
}
}
}
printSolution(dist);
}
int main() {
createGraph();
int source;
printf("Enter the source vertex: ");
scanf("%d", &source);
dijkstra(source);
return 0;
}
```

OUTPUT

Enter the number of vertices: 2

Enter the adjacency matrix:

22

22

22

54

Enter the source vertex: 5

Vertex	Distance from Source
1	999999
2	999999