10. Write a program to implement Dijkstra's algorithm to compute the shortest path through a graph.

PROGRAM:

```
INF = 1000

#  search min function
def search_min(length, se, n):
    global v
    mi = 100
    for i in range(n):
        if se[i] == 0:
            if length[i] < mi:
                mi = length[i]
                v = i
    return v


se = [0] * 16
length = []
path = []
graph = []

n = int(input("Enter No of Vertexes: "))

print("enter the adjacency matrix: ")
for i in range(n):
    graph.append(list(map(int, input().split())))

s = int(input("Enter Source node: "))

#  INTIALIZATION PART

for i in range(n):
    if graph[s][i] == 0:
        length.append(INF)
        path.append(0)
    else:
        length.append(graph[s][i])
        path.append(s)

se[s] = 1
length[s] = 0

# ITERATION PART
temp = 1
while temp:
```

```
    c = 0
    j = search_min(length, se, n)
    se.insert(j, 1)
    for i in range(n):
        if se[i] != 1:
            if graph[i][j] != 0:
                if length[j] + graph[i][j] < length[i]:
                    length[i] = length[j] + graph[i][j]
                    path[i] = j

    for i in range(1, n + 1):
        if se[i] == 0:
            c += 1
    if c == 0:
        temp = 0
    else:
        temp = 1

    # OUTPUT

print("From(sourcevertex) To ", s)
print("\tPath\t\tLength\t\t\tShortest path ")

for i in range(n):
    if i != s:
        print("\t\t%d\t\t\t%d" % (i, length[i]), end='\t')
    j = i
    while j != s:
        print("\t%d->%d" % (j, path[j]), end='\t')
        j = path[j]
    print()
```

OUTPUT:

```
Enter No of Vertexes: 4

enter the adjacency matrix:

0 6 0 1

6 0 2 4

0 2 0 1

1 4 1 0

Enter Source node: 0

From(sourcevertex) To  0
```

Path	Length	Shortest path		
1	4	1->2	2->3	3->0
2	2	2->3	3->0	
3	1	3->0		