Lab 8

IBM18CS100

```
from collections import defaultdict

class graph():

    def __init__(self):

        self.edges = defaultdict(list)
        self.weight = {}

def addEges_(self, from_node, to_node, weight):
    self.edges[from_node].append(to_node)
    self.edges[to_node].append(from_node)
    self.weight[(from_node, to_node)] = weight
    self.weight[(to_node, from_node]

def dijsktra(graph, initial, end):
    shortest_path = {initial : (None, 0)}
        current_node = initial
        visited = set()
    while current_node != end:
        visited.add(current_node)
        destinationS= graph.edges[current_node]
weight_to_current_node = shorted - paths[current_node][1]
    for next_node in destinations:
        weight = graph.weights[(current_node, next_node)]
                + weight_to_current_node
```

```
    df next_node not in shortest path
        shortest_paths [next_node] = (current_node, weight)

    else:
        Current_shortest_weight = shortest_path [next_node][0]
        if current_shortest_weight > weigh:
shortest_path [next_node] = (current_node, weight)

next_destination = {node: shortest_paths[node] for
                    node in shortest_path if node not in
                                            visited }
if not next_destination:
    return "Route not possible"
current_node = min (next_destination, key= lambda k :
                                    next_destination [k][0])

Path=[]
while current_node is not None:
        path.append (current_node)
        next_node = Shortest_path [current_node][0]
        current_node = next_node

    path = path [::-1]
    print ('Shortest Weight:', current_shortest_weight)
        print (path)
```