Name: Shreya Laddha
USN: 1BM18CS103

Cycle: 2 Lab: 3

```
from collections import defaultdict

class Graph():
    def __init__(self):
        self.edges = defaultdict(list)
        self.weights = {}

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

def dijsktra(graph, initial, end):
    shortest_paths = {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
            if next_node not in shortest path:
                shortest_paths[next_node] = (current_node,
                                             weight)
```

Name: Shreya Laddha
USN: 1BM18CS103

```
        else:
            current_shortest_weight = shortest_paths[next_node][1]
            if current_shortest_weight > weigh:
                shortest_paths[next_node] = (current_node, weight)

        next_destinations = {node: shortest_paths[node] for node in shortest_paths if node not in visited}

        if not next_destinations:
            return "Route not possible"
        current_node = min(next_destinations, key = lambda k: next_destinations[k][1])

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

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