Pramod.B.H
1BM18CS070
5-B

```
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 = shortest_paths [current_node][1]

        for next_node in destinations:
            weight = graph.weights[(current_nod, next_node)]+ weight_to_current_node
            if next_node not in shortest_paths:
                shortest_paths[next_node] = (current_node, weight)
            else:
                current_shortest_weight = shortest_paths [next_node][1]
                if current_shortest_weight > weight:
                    shortest_paths [next_node] = (current_node, weight)
```

Pramod. B. H
1BM18CS070
5-B.

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

if not next_destinations:
        return "Route Not Possible"
# next node is the destination with the lowest weight
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]
    currnt_node = next_node

path =path [::-1]
print ('shortest weighth:', current_shortest_weight)
print(path)
print("\n")

g = Graph( )
g.addEdge('a','b', 4)
g.addEdge('a','c', 2)
g.addEdge('b','c', 1)
g.addEdge('b','d', 5)
g.addEdge('c','d', 8)
g.addEdge('c','e', 10)
g.addEdge('d','e', 2)
g.addEdge('d','z', 6)
g.addEdge('e','z', 5)

dijsktra(g,'a', 'z')
dijsktra (g,'b', 'e')
```