Hrithik Agrawal
1BM18CS139

Cycle2 lab 3

```
from collections import defaultdict

class graph()
    def __init__(self):
        self.edges = defaultdict(list)
        self.weight = {}

  def addedge(self, from_node, to_node, weight):
        self.edges[from_node].append(to_node)
       self.edges[to_node].appent(from_node)
       self.weight[(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_path[current_node][i]
    for next_nod 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)
   else:
```

Hrithik Agrawal
1BM18CS134

```
        current-shortest-weight =
            shortest-paths[next-node][1]

      if current-shortest-weight > weight:
          shortest-paths[next-node]
              (current-node, weight)

    next-destinations = {node: shortest-paths[node] for
     For node in shortest paths if node
         not in visited}

  if not next-destinations:
          return "Route not possible"
    current-node = min(next-destination,
                  key=lambda k: next destination
                                          [k][1])

  path[]

  while current-nod is not None
     path.append(current-node)
        next-node = shortest-paths
                       [current-node][0]
        current-node = next-node

  path = path[::-1]

  print('shortest weights:', current-shortest-weight)
   print(path)
```