B M RAUF
22201782

LAB 06 EXPLANATION

Task 01

I solve this problem using Dijkstra's algorithm. I initialize a dictionary (distances) to store the shortest distances from the source node to each node. Initially, setting all the distances to ∞ except source node which is set to 0. I use a heapq and explore nodes based on their current shortest distance. and iteratively explore nodes and update the shortest distances to neighbouring nodes if a shorter path is found. After exploring all reachable nodes it finally return distances and print the output. If a node is not reachable it print -1 distance.

Task 02

In this task, I apply the same thing like task 01. But here I calculate shortest distances from both Alice's starting node and Bob's starting node. Then compare distances from both starting nodes to find the common node where Alice and Bob can meet in the minimum amount of time. It calculates the maximum of the two shortest distances for each node and selects the node with the minimum

maximum distance as the meeting point. If no common node is found, it returns impossible.

Task03

Same as Task 01 I use Dijkstra Algorithm and heapq for this problem too. I explore nodes to neighbouring nodes, updating the danger level of each node based on the maximum of its current danger level and the weight of the edge to its neighbour. If the recalculated danger level is lower than the previous one, I updates the distance and add the neighbour to the queue for further exploration.