**Task 1:**

The first line of the file reads the input graph's nodes and edges. The following lines show edges as triples (u, v, w) from node u to node v with weight w. The source node for shortest pathways is also read from the file. The dijkstra function in order calculates the shortest distances by maintaining a set of visited nodes, iterating through the unvisited nodes to find the one with the smallest distance from the source, marking it as visited, and updating its neighboring nodes' distances if a shorter path is found. The computed distances are written to an output file with -1 for unreachable nodes. As specified, the code efficiently uses Dijkstra's algorithm to find the shortest paths from the source node to all other graph nodes and outputs these distances.

**Task 2:**

The graph's vertex count, edge weights, and Alice (S) and Bob(T) starting nodes are read from a file. Dijkstra's approach initializes distances as infinity except for the source node and iteratively updates distances based on the minimal possible paths to find the shortest distances from it. The distances between nodes S and T and all others are determined individually. The code then finds nodes where Alice and Bob can meet quickly. It compares the minimum times of reaching each node from S or T and finds the node with the shortest time. "Impossible" if no node is detected. If a node is discovered, the function outputs and writes the minimal time and node where Alice and Bob can meet to the output file. The method calculates Alice and Bob's earliest meeting point based on node waiting times and outputs this information along with the minimum time.

**Task 3:**

The code finds the safest path from node 1 to node N in a network with weighted edges expressing relative risk. The number of graph vertices and edges is read from a file. The graph is adjacency-list. Dijkstra's algorithm is used to find the minimal danger level path to the destination node using the dijkstra function. All nodes save the starting node 1 start with infinity distances. The algorithm then iterates across nodes, adjusting the minimum danger level to reach each node based on the path's maximum danger level. The least danger level to reach node N is printed and sent to the output file by the code. If it can't reach the destination, it outputs "Impossible" and saves it to the output file. The method efficiently uses Dijkstra's algorithm to find the safest path by considering the maximum danger level, creating the output file.