Task 1:
Firstly I imported heapq to use the priority queue to efficiently explore the graph and find the shortest path. To implement the Dijkstra algorithm here I took a dictionary called distances to store the shortest distance from the start. The priority queue while loop is used to keep track of the nodes which will be explored next. The if condition checks whether the current_distance is shortest or not, depending on that the loop will continue. In the next condition if the distance is shorter than before it will assign that distance into the distances dictionary. This process will continue up until the end of priority_queue. Finally the correct distance will be returned.

Task 2:
The pq_S, pq_T while loop part is similar to task 1. Then the for loop iterates all nodes and calculates the maximum distance. It updates min_time and meeting_node variable if a smaller total_time if found. Finally based on the those value the if condition write the final output.

Task 3:
Here the Dijkstra algorithm is similar to task 1. The Dijkstra function returns the distances. In the function there is a little difference instead of keeping track of the minimum distance it keeps track of the maximum danger level encountered in the path so far. The goal is to minimize the maximum danger level. Finally the if condition write the output depending on min_danger_level.