Question 2

- The first step was to create an adjacency list for the graph. However in this list, each node needs to store the calculated distance from Syrjälä for the implementation of Dijsktra's Algorithm.
- Let's come to the main function first. The loop simply creates the adjacency list by adding edges between u-1 and v-1 since the question is 1-indexed. The edge also contains a weight w which is the distance between the flight.
- In contrast to the prior question, this is a unidirectional graph. So instead of adj[u] and adj[v] having the same values, they differ in this case as evident in the addEdge() function;
- Lastly, the main function creates an object of the Graph class and calculates the shortest distance to each city in the ans[] vector.
- Now coming to the Graph class, it basically implements the Dijsktra's algorithm using a priority queue.
- Differing to the adj list which had the type <int,long> the queue has <long, int>. This is due to the fact that the priority queue compares the first value in all the nodes. Since our weights are of the type long, we need to make this change.
- The distances are stored in the dist[] vector and we initialize dist[0]=0 because the distance from Syrjälä to Syrjälä is 0.
- Now if the queue is non empty, we iterate through the neighbors of the top element. We get the node and its assigned distances in v and w. Then upon making a

simple check if the new distance is less than the previous one, we assign the new one and push the pair into the queue.

- The if condition before the loop prevents unnecessary loop execution thus preventing TLE in 2 of the cases.
- Once the queue is empty, we have the minimum distances of each city and we can return the dist[] array.