Solution Guidelines

The task consists of several subtasks targeting various skills, from development to quality assurance. You don't necessarily have to solve everything; you can submit any subset of these subtasks that you decide. For example, you can choose to only submit the Quality Assurance subtask.

For the development subtasks, you can use any of the following languages: C, C++, Java, Python, PHP, NodeJS.

For the quality assurance subtask, you can submit your test cases in any structured format that you like: e.g., Gherkin, Excel, pseudo-code, or even a well-structured text file.

Task Statement

Background

Alice has just completed her university exams and wants to celebrate by taking a beach holiday. Unfortunately, she lives far from the seaside and must travel by plane to reach her desired destination. This journey may require her to take multiple flights due to the distance involved. As a student, Alice is on a budget. On one hand, she doesn't want to pay too much for the trip. On the other hand, she wants a comfortable journey with as few layovers as possible.

Task Objective

Your task is to help Alice find the list of all possible routes she can take. All routes must start from her current location and end at her target destination, using one or more flights. Each route must include its total price and the list of cities it passes through. The list of routes should be sorted in ascending order by price.

Input Description

You are given the following information:

- A list of direct flights between cities. Each flight is represented by its departure and arrival cities. Each city is identified by a unique three-letter code (e.g., SOF for Sofia, LON for London, FRA for Frankfurt).
- The price of each flight, as a positive integer.
- Alice's starting location (origin).
- Alice's desired holiday destination.

Constraints

- Flights are strictly one-way, meaning travel is only possible from the origin city to the destination city.
- Assume flights are always available; Alice can board any flight from her current location.
- The price of each flight is constant.
- A route cannot contain the same city more than once.

Subtask 1 - Development

Write a function that returns a list of all possible routes from the origin to the destination. The function should take three inputs – the list of flights in the network (including their respective prices), the origin, and the destination. The function should return a list of routes, sorted by price in ascending order. Each route should include the list of cities and its total price.

You can assume that the available flights and their prices are static and can be read from a file when the application starts.

Example 1

input	Output
`Flights:`	`SOF,IST,CMB,MLE,70`
`  SOF,IST,10`	
`  IST,CMB,20`	
`  CMB,MLE,40`	
`Origin: SOF`	
`Destination: MLE`	

Example 2

Input	Output
`Flights:`	`SOF,LON,MLE,30`
`  SOF,MLE,70`	`SOF,MLE,70`
`  SOF,LON,10`	
`  LON,MLE,20`	
`Origin: SOF`	
`Destination: MLE`	

Example 3

Input	Output
`Flights:`	`No routes`
`  SOF,LON,10`	
`  SOF,NYC,20`	
`Origin: SOF`	
`Destination: MLE`	

Subtask 2 - Development

Write an HTTP service exposing a simple REST JSON API that can help Alice explore different destinations. The API should expose a single POST endpoint that accepts an origin and a destination. The response should include a list of routes sorted by price in ascending order.

Example 4

Request	Response
{	[
”origin”: ”SOF”,	{
”destination”: ”MLE”	”route”: [”SOF”, ”LON”, ”MLE”],
}	”price”: 30
	},
	{
	”route”: [”SOF”, ”MLE”],
	”price”: 70
	}
	]

Subtask 3 - Development

Add an additional parameter to the API – maxFlights, which limits the number of flights in a route. A maxFlights value of 1 means that the response should only include direct flights. This parameter should be optional; if omitted, there is no limit on the number of flights.

Example 5

Request	Response
{	[
”origin”: ”SOF”,	{
”destination”: ”MLE”,	”cities”: [”SOF”, ”MLE”],
”maxFlights”: 1	”price”: 70
}	}
	]

Subtask 4 – Quality Assurance

Define comprehensive test cases to validate the correctness of the API defined in Subtasks 2 and 3. Don’t just focus on a few straightforward cases; try to cover all edge cases you can think of.