Mini assignment

Summary

Implement a data structure to manage bookings at an airline

Background

Passengers (“pax” for short) book tickets at an airline to take them from origin to destination. Some bookings are non-stop, while others go through intermediate airports (“layovers”). Airports can be uniquely identified by their three-letter IATA code. For example, Amsterdam Airport Schiphol is AMS, Atlanta is ATL.

Below are examples of bookings:

|  |  |  |
| --- | --- | --- |
| Pax name | Departure (UTC) | Itinerary |
| Alice | May-26 06:45 2020 | LHR→AMS |
| Bruce | Jun-04 11:04 2020 | GVA→AMS→LHR |
| Cindy | Jun-06 10:00 2020 | AAL→AMS→LHR→JFK→SFO |
| Derek | Jun-12 08:09 2020 | AMS→LHR |
| Erica | Jun-13 20:40 2020 | ATL→AMS→AAL |
| Fred | Jun-14 09:10 2020 | AMS→CDG→LHR |

Requirements

* Implement a data structure for bookings such that you can efficiently:
* add bookings
* select bookings departing before a given time (e.g. Jun 12 12:00 2020)
* select bookings visiting two airports sequentially. (e.g. the search AMS→LHR gives you bookings with itineraries like HAM→AMS→LHR, AMS→LHR, AAL→AMS→LHR→JFK→SFO, etc. but not LHR→AMS or AMS→CDG→LHR, for example. )
* Leave the code in a state that reflects your standard for production-grade quality. We score your solution based on:
* simplicity (the solution should be no more complex than needed)
* testability and reusability (another developer should be able to continue with your work)
* use of appropriate types and data structures
* Implement the assignment in the specified programming language. Preferably without using third-party packages.
* Candidates typically finish the assignment within 1,5 hours. You do *not* need to include database interactions, HTTP API, file I/O, logging, command-line interface, user interface, pretty display, concurrency, deployment configuration, or parsing functionality.