Code Challenge: Booker

Intro

Avani Labs is a company that allows clients to perform tests on their sites and can even provide some test vehicles.
They have multiple buildings, and each Building can be separated in different sections and have any number of vehicles.

Issue

Sometimes the agents might have communication issues and end up booking a resource that is already reserved.
For example, the seller Victor booked the building "Warehouse A" to Volvo Trucks to the day 2029/10/01.
But Carla had already booked the whole "Pearl Sector", which is part of "Warehouse A", for the same day.

Solution

To fix that, they need a tool to help them manage their resources so an operator can insert a list of queries and get an output showing if the reservations are conflicting or not.

You, as a capable developer, are assigned to create Booker, a program that can handle that.

Inputs

The program will process two files to check for conflicts:

Resources file:

A json formatted file that describes the company resources and obeys the following schema:

- Resources: A collection of entities that includes buildings. It is the top-level object in the schema.
    - Buildings: An array of building objects. Buildings can have any number of sections and vehicles, including none. A building cannot contain other buildings.
        - id: A string representing the unique identifier for the building.
        - sections (optional): An array of section objects within a building.
            - id: A string representing the unique identifier for the section.
            - sections (optional): An array of nested section objects within another section.
            - vehicles (optional): An array of strings representing vehicle identifiers within a section. A vehicle can be part of a section,

but it does not need to be. A vehicle can also be shared by multiple sections but not by multiple buildings.

- vehicles (optional): An array of strings representing vehicle identifiers within a building. Vehicles must be within buildings.

The file `Example/resources.json` contains an example of this schema.

The file `Example/resources.jpg` illustrates the contents of the example json file. (This is just an example image; it shall not be processed by Booker)

Queries file:

A series of lines with queries to manage bookings, following this format:

Comments:

Lines beginning with a '#' symbol are comments and are ignored by the parsing program.

Blank Lines:

Blank lines are also ignored and do not affect the parsing.

Commands:

Each line contains a command related to booking or checking the status of a resource, following one of three specific formats:

`book <RESOURCE_ID> <DATE>`

Purpose: To book a resource (and all resources within it) for the given date.
Return: "ok" if the booking is successful, "failed" if unable to book the resource.

`is_booked <RESOURCE_ID> <DATE>`

Purpose: To check if a resource (or all resources within it) is booked for a given date.
Return: "yes" if the resource is booked, "no" otherwise.

`is_available <RESOURCE_ID> <DATE>`

Purpose: To check if a resource is available to be booked for a given date.
Return: "yes" if the resource is available, "no" otherwise.

Parameters: RESOURCE_ID is the identifier for the resource to check, and DATE is the date to query in YYYY/MM/DD format.

The file `Example/queries.txt` contains an example that matches the example resource description file.

Output

The given inputs should be processed and the results for the queries should be output to a text file. The output lines should be in the same sequence as the queries that generated these results.

The file `Example/results.txt` contains an example that matches the example resource description and queries files.

Arguments

The program should receive 3 paths as arguments: resources, queries, and results.

For example:

```
./booker inputs/resource.json inputs/queries.txt outputs/results.txt
```

Notes

Here are some general notes about the program:

- Buildings, Sections, and Vehicles are considered resources.
- A resource cannot be booked for a specific date if it is already booked for this day.
- A resource cannot be booked for a specific date if any of the resources within it cannot be booked.
- The application should finish after generating the output file.

Expected

This is what we expect from you:

- The source code for the application on the programming language you consider is most appropriate.
- Instructions on how to build and run the code.
- General Documentation about the project containing relevant UML diagrams.
- If you feel that tests are relevant, test cases and instructions on how to run them.

Feel free to point out and briefly explain any details or efforts you find relevant to your implementation, such as reducing time complexity, for example.
Justifications of decisions made are also appreciated.

You can ask any questions you may have.

We hope you have fun with that challenge!