Winter Wares at Refrostly

This project is a small version of a problem that one might encounter at Remitly. For this project, we are interested in seeing how you solve problems and build software, so please treat this problem as you would a deliverable for work. In other words, your solution should be considered "production quality". It is designed to take a few hours to complete.

Feel free to use any language that you are comfortable with to solve the problem described, but keep in mind that you will need to provide us with instructions for any special steps to build and deploy your solution.

Take your time and have fun with this problem, and keep in mind that there is no 'perfect' solution. We will be happy to answer any further questions you have about the problem.

Problem:

The online store Refrostly sells the following products:

- Skis
- Shovels
- Sleds
- Snowblowers
- Winter tires

When a customer places an order, Refrostly calls up the manufacturers, orders the products from manufacturers, who ships them to Refrostly's warehouse. Refrostly then sends them to the customer.

In order to speed up delivery times and make their service cheaper, Refrostly is experimenting with stocking their warehouse *before* customers order the inventory, so it's ready to ship as soon as the customer orders it.

Refrostly's Inventory Analysis team has come up with a Restocking Algorithm, which they've set up to generate a list of inventory purchases. The team would like to test this algorithm against the order history from last year, to see if it can generate effective restocking instructions for the coming year.

Refrostly needs you to create an application that evaluates a restocking algorithm against Refrostly's actual order history from the past year. By keeping track of when Refrostly sells its inventory (customer order events), and when it would restock its warehouse with new inventory (inventory restocking events), this application will track a running inventory of all of Refrostly's products.

Input:
To test the strategies, the Analysis team is presenting you with a set of files:

1. Refrostly's 2018 customer order events (orders.json), with the following data:
    - Customer ID
    - Order ID
    - Date and time of the order
    - Item ordered
    - Quantity of item
    - Price that Refrostly charged per item

2. A list of inventory restocking events generated by the Restocking Algorithm (restocks.json)
    - Date and time of the restock
    - Item stocked
    - Quantity of item
    - Manufacturer name
    - Price that Refrostly paid per item

    Feel free to use the provided sample files during development.

Output:

1. Whether the restocking algorithm ended in **SUCCESS** or **OUT OF STOCK**. If customers order more inventory than Refrostly actually has in the warehouse at any given time, then our restocking algorithm would cause us to run **out of stock** and turn away customers. If we never run out of stock and finish processing all customer order events, then the restocking algorithm is a **success**.
2. If the restocking algorithm is a **SUCCESS**, output the remaining inventory.
3. If the restocking algorithm leads to **OUT OF STOCK**, output which product Refrostly ran out of, and when it ran out.

For example, with the two sample files provided, the application should explain that the restocking algorithm was a **SUCCESS**, with the following extra inventory in the warehouse:

- 4 shovels
- 4 snowblowers
- 2 tires
- 1 sled

Deliverables:
When complete, please email your solution to noamc@remitly.com. It should contain:

1. A .zip archive (please include your .git directory so we can read your commits) or link to a Github repository with the project source code
2. A README file with a description of your application, including design decisions and assumptions made, as well as instructions for running the application