Background

Zipline both flies autonomous vehicles and operates a full-fledged logistics system. We run operations out of our distribution centers, which we call "nests." At a given nest, we have an inventory of medical supplies and a team of operators who manage that inventory and process orders from doctors.

We’re going to build a lightweight inventory management and order processing system. This system will

- manage an inventory of products
- keep track of which items have been stocked
- transforming incoming orders into outgoing shipments as appropriate

There are a number of reasons that we might not be able to directly ship the order as received:

- The order is too big to fit in a single shipment (our max package size is 1.8kg)
- We could be out of the requested product, in which case the order may be only partially filled, and remaining items would not be able to be shipped until a restock happens

Problem

Let's build the backend of the system. Work in your best language (Python preferred). In the interest of respecting your time, we ask that you spend no more than **two hours** total (this time may be split up over multiple working sessions).

- Assume that the system starts with an "empty" inventory.

- Before anything else happens, we will call **init_catalog**. This serves to capture the list of products that are allowed to be stocked or ordered.

- After this, our system will receive *inputs* via two interfaces: **process_order**, or **process_restock**. These represent APIs that would be invoked by an external service.

    - **process_restock** is called to add new inventory to the system

    - **process_order** will be called to inject a new order into the system

- Our goal is to make sure that, eventually, our program invokes a **ship_package** API as many times as necessary to fulfill an order.

- The total products shipped should eventually equal the total products requested in the original order, but no single call to **ship_package** should ever exceed the maximum shipment size (1.8kg).
    - Remember, if we don't currently have enough inventory in stock, the call to **ship_package** may need to be deferred until after a future call to **process_restock**.

For convenience, each of these functions can be assumed to take a native list or dictionary loaded from a JSON payload.

- **init_catalog(product_info)** – this will take an input *product_info* manifest and do any setup necessary. This will be called once at the beginning program before any calls to **process_restock** or **process_order**. After **init_catalog** is first called, we can assume we have 0 of each product type.

  Example JSON for product_info:

[{"mass_g": 700, "product_name": "RBC A+ Adult", "product_id": 0}, {"mass_g": 700, "product_name": "RBC B+ Adult", "product_id": 1}, {"mass_g": 750, "product_name": "RBC AB+ Adult", "product_id": 2}, {"mass_g": 680, "product_name": "RBC O- Adult", "product_id": 3}, {"mass_g": 350, "product_name": "RBC A+ Child", "product_id": 4}, {"mass_g": 200, "product_name": "RBC AB+ Child", "product_id": 5}, {"mass_g": 120, "product_name": "PLT AB+", "product_id": 6}, {"mass_g": 80, "product_name": "PLT O+", "product_id": 7}, {"mass_g": 40, "product_name": "CRYO A+", "product_id": 8}, {"mass_g": 80, "product_name": "CRYO AB+", "product_id": 9}, {"mass_g": 300, "product_name": "FFP A+", "product_id": 10}, {"mass_g": 300, "product_name": "FFP B+", "product_id": 11}, {"mass_g": 300, "product_name": "FFP AB+", "product_id": 12}]

- **process_order(order)** – This represents an incoming order from a hospital. The argument is a description of the order to process. We should eventually invoke **ship_package** (multiple times if necessary) such that all of the products listed in the order are shipped, but we should never ship products that have not yet been stocked as part of our inventory. Remember that each package has a maximum mass of 1.8 kg.

  Example JSON for order

{"order_id": 123, "requested": [{"product_id": 0, "quantity": 2}, {"product_id": 10, "quantity": 4}]}

- **process_restock(restock**) – A restock happens when we receive new products that must be added to the inventory. This API call provides the list of products that have been supplied. Additionally, if there are any pending orders (or partial orders), which were not previously shipped, they should be shipped immediately as a result of this function call.

Example JSON for restock

[{"product_id": 0, "quantity": 30}, {"product_id": 1, "quantity": 25}, {"product_id": 2, "quantity": 25}, {"product_id": 3, "quantity": 12}, {"product_id": 4, "quantity": 15}, {"product_id": 5, "quantity": 10}, {"product_id": 6, "quantity": 8}, {"product_id": 7, "quantity": 8}, {"product_id": 8, "quantity": 20}, {"product_id": 9, "quantity": 10}, {"product_id": 10, "quantity": 5}, {"product_id": 11, "quantity": 5}, {"product_id": 12, "quantity": 5}]

- **ship_package(shipment)** – This is just a stub API. In reality, this would need to feed a user-interface informing a fulfillment operator to pack and ship the package; in our case, all this needs to do is print out the shipment to the console.

  Example JSON for shipment

{"order_id": 123, "shipped": [{"product_id": 0, "quantity": 1}, {"product_id": 10, "quantity": 2}]}