```
                    _     _
                   /\   /\___ (_) __| |
                   \ \ / / _ \| |/ _` |
                    \ V / (_) | | (_| |
                     \_/ \___/|_|\__,_|

                                    _
  /\  /\__ _ _ ____   _____  ___| |_ ___ _ __
 / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__|
/ __  / (_| | |   \ V /  __/\__ \ ||  __/ |
\/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|
```

Version 1.0 by Andres Arias

Introduction

Welcome to the Void Harvester project! The year is 2151. You have been selected to work for the world's first space mining company: Void Harvester. Void Harvester's investors have expressed their wish for trading software to be made available. Void Harvester, or VH for short has designated a product owner to create several user stories. These define the roadmap for the up and coming web application.

My name is Andres Arias and I will be the product owner for this project. If there are any questions please feel free to contact me on anar@ageras.com. I expect that the work can be done in around 4 hours. Please do not use public source version control repositories since this project is confidential. Treat it as if it would be a real assignment.

Good luck!

Business Rules

Asteroids exist in 4 types. Each type has a different composition and therefore a different yield as outlined in the table below:

Type	Nickel	Iron	Cobalt	Water	Nitrogen	Ammonia
C	0.75%	2.00%	0.25%	83.00%	8.50%	5.50%
S	10.06%	20.49%	35.57%	0.34%	8.07%	25.50%
M	24.95%	20.30%	20.24%	11.02%	23.12%	00.37%
V	0.76%	2.62%	11.47%	1.51%	31.39%	52.25%

An asteroid will not be productive on the first day. It will take time to setup mining equipment and ship in the required crew. The time this takes is dependent on the mass of the asteroid. The formula is as follows, with *t* in *days*:

```
t = log(m)
```

Once the mining has started the daily yield can be represented by, where f is the remaining decimal fraction of mass of the asteroid and m is the remaining mass of the asteroid:

for f >= 0.02:

```
yield = (0.04 + ln(f) / 100) * m
```

for f < 0.02

```
yield = 0
```

Story 1: As a trade manager I can load asteroids into the store to calculate production

When the system starts it expects a *yaml* file to be specified as a command line argument. This *yaml* file contains the asteroids that AH is mining. The mass is specified in millons of tons. The system should print a summary of the loaded asteroids to *stdout*.

Example file:

```
-   name: Vesta
    type: C
```

```
    mass: 60.16
-   name: Ceres
    type: S
    mass: 107.52
-   name: Psyche
    type: M
    mass: 25.04
-   name: Bennu
    type: V
    mass: 83.91
```

Story 2: As a customer I can ask the current stock on a particular day

The trade manager as well as prospective customers need to know what amount of resources is at hand at any given time. Create an API call as follows:

```
GET /resources/<n>
```

n is the day that the stock is being queried for. This will return the stock as follows for n = 10:

```
{
    "nickel": 4.07,
    "iron": 6.72,
    "cobalt": 11.71,
    "water": 13.13,
    "nitrogen": 10.85,
    "ammonia": 17.25
}
```

Story 3: As a customer I can place an order

The customer needs to be able to place and order on a specific date. Amount of each resource can be specified

```
PUT /order/<order_id>
```

The customer supplies the order id which should be a UUID. The customer also supplies the amount of resources he wants to order, and when he wants to order, for example:

```
{
    "day": 10,
    "nickel": 5,
    "iron": 10,
    "cobalt": 2,
    "water": 15,
    "nitrogen": 15,
    "ammonia": 12
}
```

The amounts are in millions of tons. When an order cannot be completely fulfilled it will be partially fulfilled.