Introduction

At Carta, we help companies manage and understand **ownership**. Corporations represent ownership on a ledger called a *capitalization table*, or "cap table" for short. This table lists each of the company's shareholders (investors, employees, and advisors), the number of shares they own, the price they paid to purchase them, and their ownership percentage. Keeping a cap table accurate and up to date is a challenging problem, and most cap tables are broken as a result.

When a company grants an **equity award** (shares) to an employee, it usually has to **vest** before you can do anything with it. This vesting usually occurs in increments over a period of time, encouraging recipients to stay longer at the company. A **vesting schedule** defines how much total equity has been vested over a time.

In this exercise, you will generate a cumulative vesting schedule from a series of individual **vesting events**.

Requirements

Write a command-line program that reads in a file of **vesting events** and outputs a **vesting schedule** to `stdout`. An automated tool will be used to validate the correctness of your submission, so strict adherence to this specification is necessary.

- The program should receive a positional **filename** argument and read event data from that file.
- The program should receive a second positional **target date** argument and calculate the total number of shares vested **on or before** that date.

```
> ./vesting_program example.csv 2020-03-03
```

- If using python, the program **must not** use the pandas library. While using pandas is a reasonable way to solve this problem, it generally does not provide informative signals for the **Evaluation Criteria**.

Iterative Exercise

This exercise is broken down into multiple stages. **This exercise is designed to be completed in two hours, but you may take as much time as you need.**

Evaluation Criteria

Your submission will be evaluated on the following axes:

- Is the output **correct**?
- Is the code **clear** and **readable**?
- Does it exhibit good **separation of concerns**?
- Do methods and classes follow the **single responsibility principle**?
- Is the code **maintainable** and easy to keep free from **side-effects**?
- Is the code appropriately **extensible**?
- Does the code handle large data sets **robustly**?

Stage 1

Input data is CSV and includes **vesting events** that indicate how many shares vest from each **equity award** on each date. Input data **does not** include a header row.

```
VEST,<<EMPLOYEE ID>>,<<EMPLOYEE NAME>>,<<AWARD ID>>,<<DATE>>,<<QUANTITY>>
```

```
> cat example1.csv
VEST,E001,Alice Smith,ISO-001,2020-01-01,1000
VEST,E001,Alice Smith,ISO-001,2021-01-01,1000
VEST,E001,Alice Smith,ISO-002,2020-03-01,300
VEST,E001,Alice Smith,ISO-002,2020-04-01,500
VEST,E002,Bobby Jones,NSO-001,2020-01-02,100
VEST,E002,Bobby Jones,NSO-001,2020-02-02,200
VEST,E002,Bobby Jones,NSO-001,2020-03-02,300
VEST,E003,Cat Helms,NSO-002,2024-01-01,100
```

The **output** includes, per-award, the total shares vested on or before the **target date**:

```
> ./vesting_program example1.csv 2020-04-01
E001,Alice Smith,ISO-001,1000
E001,Alice Smith,ISO-002,800
E002,Bobby Jones,NSO-001,600
E003,Cat Helms,NSO-002,0
```

ˇ

- Include all employees and awards in the output, even if no shares are vested by the given date
- Output should be **ordered** by Employee ID and Award ID

Stage 2

Input data also includes **cancellation events** that indicate how many shares were cancelled from each **equity award** on each date. Input data **does not** include a header row.

```
CANCEL,<<EMPLOYEE ID>>,<<EMPLOYEE NAME>>,<<AWARD ID>>,<<DATE>>,<<QUANTITY>>
```

```
> cat example2.csv
VEST,E001,Alice Smith,ISO-001,2020-01-01,1000
CANCEL,E001,Alice Smith,ISO-001,2021-01-01,700
```

The **output** should now subtract all shares cancelled on or before the **target date**:

```
> ./vesting_program example2.csv 2021-01-01
E001,Alice Smith,ISO-001,300
```

- Valid cancellation events will never exceed the number of shares vested on or before that date for that equity award.

Stage 3

Input data now specifies the quantity of shares in **fractional** values.

```
> cat example3.csv
VEST,E001,Alice Smith,ISO-001,2020-01-01,1000.5
CANCEL,E001,Alice Smith,ISO-001,2021-01-01,700.75
VEST,E002,Bobby Jones,ISO-002,2020-01-01,234
```

The program must now accept a third optional positional argument indicating how many digits of precision to recognize from the **input** and include in the **output**. Valid values are between 0 (default) and 6.

- Inputs with less precision than the specified precision should be supported.

- Inputs exceeding this precision should be **truncated** or **rounded down**.
- For example, with 2 digits of precision specified:

  treat `100.5` as `100.50`
  treat `100.4567` as `100.45`

The **output** should now contain fractional quantities with the given precision:

```
> ./vesting_program example3.csv 2021-01-01 1
E001,Alice Smith,ISO-001,299.8
E002,Bobby Jones,ISO-002,234.0
```

Submission

While there is no time limit for the assignment, it is designed to be completed in one sitting.

Submit your code as a zip file (no GitHub links, please!) using the submission link provided by your recruiter. Include your initials in the filename, e.g. `AB-CARTA.zip` or `AB-CARTA.tar.gz`

Your Submission also requires a `README` explaining how to build and run your code, and a few sentences describing any key design decisions and documenting any assumptions you made. **If a requirement is unclear, make a choice based on your understanding of the requirement and document your assumption.** If applicable, add a short description of what changes you would have made if there were no time constraints.

So we can ensure that your assignment remains anonymous during review, please **do not** include your name *anywhere* in your code or in your `README`.

- Do **NOT** include your name in anything you submit.
- Do **NOT** include compiled binaries in your submission.