

Introduction

At Carta, we help companies manage and understand **ownership**. When a company grants an **equity award** 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.

In this exercise, you'll build a command-line program that generates a cumulative **vesting schedule** from a series of individual vesting events. The exercise is designed to be completed in **approximately 2 hours**, though you may take as much time as you need.

Before You Start

Time Expectation

- **Designed for:** 2 hour
- **You may take:** As much time as you need
- **Stages:** 3 incremental stages that build on each other

Submission Format

- **Package your code** as a zip file: **AB-CARTA.zip** (use your initials)
- Must include:
 - All source code files
 - **README** with:
 - How to build and run your code
 - Any key design decisions you made
 - Assumptions you made (if requirements were unclear)
 - If there were time constraints, note what you would change with more time
 - Any LLM prompts you used to build your solution
- **Submit** using the link provided by your recruiter

Anonymity Requirements

To ensure fair review, your submission must be anonymous:

- **Do NOT** include your name anywhere in code, comments, or README
- **Do NOT** include compiled binaries in your submission

Core Requirements

Write a command-line program that can be run as follows:

```
> ./vesting_program <filename> <target_date> [precision]
```

An automated validation tool will verify your submission so strict adherence to the specifications is required.

Required Arguments

- **filename**
 - Type: Positional (1st)
 - Description: CSV file containing vesting events
 - Example: `example.csv`
- **target_date**
 - Type: Positional (2nd)
 - Description: Calculate total shares vested on/before this date
 - Example: `2020-03-03`
- **precision**
 - Type: Positional (3rd), Optional
 - Description: Number of decimal digits to handle for both input and output (default: 0).
 - Example: `2`

Input Format

- CSV file with **no header row**

Output Format

- Print to **stdout** (standard output)
- One line per employee + award combination
- Format:
`<EMPLOYEE_ID>, <EMPLOYEE_NAME>, <AWARD_ID>, <TOTAL_SHARES_VESTED>`
- Must be **ordered by**: Employee ID (ascending), then Award ID (ascending)

Python Constraint

If using Python, you **must NOT** use the `pandas` library. While pandas is reasonable for this problem, it doesn't provide informative signals for our evaluation criteria.

Multi-Stage Exercise

This exercise builds incrementally across 3 stages.

You should:

1. Complete Stage 1 fully
2. Modify your code to handle Stage 2 requirements
3. Further modify your code to handle Stage 3 requirements

Your final submission should handle all three stages. Each stage introduces new complexity:

- **Stage 1:** Basic vesting events
- **Stage 2:** Cancellation events (subtract shares)
- **Stage 3:** Fractional shares with configurable precision

Stage 1: Basic Vesting

Input Data Format

CSV with vesting events showing shares vested from each equity award:

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

Example (`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
```

Expected Output

Running:

```
./vesting_program example1.csv 2020-04-01
```

Should output:

```
E001,Alice Smith,ISO-001,1000
E001,Alice Smith,ISO-002,800
E002,Bobby Jones,NSO-001,600
E003,Cat Helms,NSO-002,0
```

How This Works

For Alice Smith, Award ISO-002 as of 2020-04-01:

- 2020-03-01: VEST 300 shares → Running total: 300
- 2020-04-01: VEST 500 shares → Running total: 800
- **Output: 800 shares**

For Cat Helms, Award NSO-002 as of 2020-04-01:

- 2024-01-01: VEST 100 shares (future date, after target)
- **Output: 0 shares**

Stage 1 Requirements

- Include **all employees and awards** in the output, even if 0 shares vested by the target date
- Output must be **ordered by** Employee ID, then Award ID

Stage 2: Cancellation Events

Your program must now also handle cancellation events that subtract shares.

Input Data Format

The CSV input now also includes:

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

Example ([example2.csv](#)):

```
VEST,E001,Alice Smith,ISO-001,2020-01-01,1000  
CANCEL,E001,Alice Smith,ISO-001,2021-02-01,700
```

Expected Output

Running:

```
./vesting_program example2.csv 2021-02-01
```

Should output:

```
E001,Alice Smith,ISO-001,300
```

How This Works

For Alice Smith, Award ISO-001 as of 2021-02-01:

- 2020-01-01: VEST 1000 shares → Running total: 1000
- 2021-02-01: CANCEL 700 shares → Running total: 300
- **Output: 300 shares**

Stage 2 Requirements

- Subtract cancelled shares from vested total on or before the target date
- **Validation rule:** The sum of shares cancelled on a given day must not exceed the sum of shares vested on or before that day. Any cancellation event that exceeds the vested amount is invalid.

Stage 3: Fractional Shares & Precision

Your program must now accept an optional precision argument and handle fractional share quantities.

New Optional Argument: Precision

The program now accepts a third positional argument for decimal precision:

```
./vesting_program <filename> <target_date> [precision]
```

This precision argument indicates how many digits of precision **to recognize from the input and to display in the output**.

- **Valid range:** 0 to 6 (inclusive)
- **Default:** 0 (if not specified)
- Inputs and outputs with less precision than specified should be supported
- Inputs and outputs exceeding specified precision should be **rounded down** (truncated)

Input Data Format

Quantities in the CSV input may now be fractional:

Example (`example3.csv`):

```
VEST,E001,Alice Smith,ISO-001,2020-01-01,1000.5
CANCEL,E001,Alice Smith,ISO-001,2021-02-01,700.75
VEST,E002,Bobby Jones,ISO-002,2020-01-01,234
```

Expected Output

Running:

```
./vesting_program example3.csv 2021-02-01 1
```

Should output:

```
E001,Alice Smith,ISO-001,299.8
E002,Bobby Jones,ISO-002,234.0
```

Precision Examples

With **no precision argument** specified:

- Input: 100.5 → Treat as: 100 (rounded down)
- Input: 100.4567 → Treat as: 100 (rounded down)
- Output: 200.012 → Treat as: 200 (rounded down)

With **2 digits of precision** specified:

- Input: 100.5 → Treat as: 100.50
- Input: 100.4567 → Treat as: 100.45 (rounded down)
- Output: 200.012 → Treat as: 200.01 (rounded down)

Stage 3 Requirements

- Support fractional quantities in input
- Accept optional third argument for precision (0-6, default 0)
- Round down (truncate) inputs and outputs exceeding specified precision

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**?

Quick Reference Checklist

Before submitting, verify:

- Handles VEST events (Stage 1)
- Handles CANCEL events (Stage 2)
- Handles fractional shares (Stage 3)
- Accepts 3 positional arguments: **filename, target_date, [precision]**
- Default precision is 0 if not specified
- Output ordered by Employee ID, then Award ID
- Includes all employees/awards even if 0 shares vested
- Does NOT use pandas (if Python)
- **README** included with build/run instructions
- Design decisions documented
- Submission is anonymous (no name anywhere)
- Submitted as zip file with correct naming

Appendix A: Background on Vesting

What is Vesting?

Corporations represent ownership on a ledger called a capitalization table ("cap table"). This table tracks:

- Each company's shareholders (investors, employees, advisors)
- Number of shares they own
- Price they paid
- Ownership percentage

Keeping a cap table accurate and up to date is a challenging problem, and [most cap tables are broken](#) as a result.

Vesting Schedules

When companies grant equity awards to employees, these awards typically vest over time.

Common vesting schedules include:

- **4-year vest with 1-year cliff:** No shares vest for the first year, then 25% vest, then monthly vesting for remaining 3 years
- **Monthly vesting:** Shares vest in equal increments each month
- **Milestone-based vesting:** Shares vest when specific goals are achieved

Vesting events define how much total equity has been vested over time. This exercise asks you to calculate cumulative vesting from a series of individual vesting events.