Code Challenge: Capital Gains

Challenge Context

You are tasked with implementing an application that calculates how much tax you should pay based on the profit or losses of a stock market investment.

Please make sure you read all the instructions below, and feel free to ask for clarifications if needed.

Sample usage of the Capital Gains

How the program should work

Input

Your program is going to receive a list of stock market operations encoded in `json` through the standard input (stdin) containing the following fields:

Field Name	Meaning
`operation`	If the operation was a `buy` or `sell`
`unit-cost`	The stock's unit cost
`quantity`	The quantity of stocks negotiated

Here is an example of input:

```
[{"operation":"buy", "unit-cost":10, "quantity": 10000}, {"operation":"sell",
"unit-cost":20, "quantity": 5000}]
```

The last line of the input will be a blank line.

Output

The program should return a list describing how much taxes should be paid for each operation received encoded in `json` through standard output (stdout), containing the following field:

Field Name	Meaning
`tax`	The amount of taxes to be paid based on operation

Here is an example of the expected output:

```
[{"tax":0}, {"tax":10000}]
```

The result list should have the exact size of the input list of operations. e.g., given three operations (buy, buy, sell), there should be three results representing the taxes for each operation.

Capital Gains Rules

The program should handle two kinds of operations (`buy` and `sell`), they should comply with the following rules:

- You do not pay any taxes for buying stocks;
- To determine if you have a profit or a loss, you should use a **weighted-average price**, for which the weight is the number of stocks you have bought with that price. e.g., if you have bought 10 stocks for R$ 20 and 5 for R$ 10, the weighted average is (10 x 20 + 5 x 10) / 15 = 16.66;
- Losses (i.e., you are selling for a price lower than the weighted average price you have bought) do not pay any taxes, and you need to deduct the loss from subsequent profits before calculating the tax;
- You do not pay any taxes if the total amount (unit cost of selling x quantity) is less than or equal to R$ 20000.00 (but remember to deduct the losses from the subsequent profits) - remember to use this and not the profit to decide whether or not you should pay taxes;
- The percentage we pay in taxes is 20% of the profit (i.e., you are selling for a higher price than the weighted average of stocks you have bought). If you have bought shares with different prices, you may use the **weighted-average price** paid (in which the weight is the quantity of stocks bought with that price);
- You should use a past loss on multiple future profits (until you deduct the entire amount);

You can assume that no operation will sell more stocks than you currently have.

Capital Gains Examples

Case #1

Operation	Unit Cost	Quantity	Tax	Explanation
`buy`	10	100	0	Buying stocks do not pay taxes
`sell`	15	50	0	Total amount less than R$ 20000
`sell`	15	50	0	Total amount less than R$ 20000

Input:

```
[{"operation":"buy", "unit-cost":10, "quantity": 100},{"operation":"sell", "unit-cost":15, "quantity": 50},{"operation":"sell", "unit-cost":15, "quantity": 50}]
```

Output:

```
[{"tax": 0},{"tax": 0},{"tax": 0}]
```

Case #2

Operation	Unit Cost	Quantity	Tax	Explanation
`buy`	10	10000	0	Buying stocks do not pay taxes
`sell`	20	5000	10000	Profit of R$ 50000: 20% is R$ 10000 - no previous losses to use
`sell`	5	5000	0	Loss of 25000: no tax

Input:

```
[{"operation":"buy", "unit-cost":10, "quantity": 10000},{"operation":"sell",
"unit-cost":20, "quantity": 5000},{"operation":"sell", "unit-cost":5, "quantity":
5000}]
```

Output:

```
[{"tax": 0},{"tax": 10000},{"tax": 0}]
```

Case #3

Operation	Unit Cost	Quantity	Tax	Explanation
`buy`	10	10000	0	Buying stocks do not pay taxes
`sell`	5	5000	0	Loss of R$ 25000: no tax
`sell`	20	5000	5000	Profit of R$ 50000 - deduct Loss of R$ 25000: 20% of R$ 25000 -> R$ 5000

Input:

```
[{"operation":"buy", "unit-cost":10, "quantity": 10000},{"operation":"sell",
"unit-cost":5, "quantity": 5000},{"operation":"sell", "unit-cost":20, "quantity":
5000}]
```

Output:

```
[{"tax": 0},{"tax": 0},{"tax": 5000}]
```

Case #4

Operation	Unit Cost	Quantity	Tax	Explanation
`buy`	10	10000	0	Buying stocks do not pay taxes
`buy`	25	5000	0	Buying stocks do not pay taxes
`sell`	15	10000	0	Considering average price (R$ 15) -> no profit or loss

Input:

```
[{"operation":"buy", "unit-cost":10, "quantity": 10000},{"operation":"buy",
"unit-cost":25, "quantity": 5000},{"operation":"sell", "unit-cost":15,
"quantity": 10000}]
```

Output:

```
[{"tax": 0},{"tax": 0},{"tax": 0}]
```

Case #5

Operation	Unit Cost	Quantity	Tax	Explanation
`buy`	10	10000	0	Buying stocks do not pay taxes
`buy`	25	5000	0	Buying stocks do not pay taxes
`sell`	15	10000	0	Considering average price (R$ 15) -> no profit or loss
`sell`	25	5000	10000	Considering average price (R$ 15) -> R$ 50000 of profit; 20% of R$ 50000 is R$ 10000

Input:

```
[{"operation":"buy", "unit-cost":10, "quantity": 10000},{"operation":"buy",
"unit-cost":25, "quantity": 5000},{"operation":"sell", "unit-cost":15,
"quantity": 10000},{"operation":"sell", "unit-cost":25, "quantity": 5000}]
```

Output:

```
[{"tax": 0},{"tax": 0},{"tax": 0},{"tax": 10000}]
```

Case #6

Operation	Unit Cost	Quantity	Tax	Explanation
`buy`	10	10000	0	Buying stocks do not pay taxes
`sell`	2	5000	0	Loss of R$ 40000: total amount is less than R$ 20000, but we should deduct that loss regardless of that
`sell`	20	2000	0	Profit of R$ 20000: if you deduct all the loss, it is zero (and you still have R$ 20000 to deduct)
`sell`	20	2000	0	Profit of R$ 20000: if you deduct all the loss, it is zero (and you now have zero to deduct)
`sell`	25	1000	3000	Profit of R$ 15000 and zero losses you can deduct: tax is 20% of that

Input:

```
[{"operation":"buy", "unit-cost":10, "quantity": 10000},{"operation":"sell",
"unit-cost":2, "quantity": 5000},{"operation":"sell", "unit-cost":20, "quantity":
2000},{"operation":"sell", "unit-cost":20, "quantity": 2000},{"operation":"sell",
"unit-cost":25, "quantity": 1000}]
```

Output:

```
[{"tax": 0},{"tax": 0},{"tax": 0},{"tax": 0},{"tax": 3000}]
```

Application State

The program **should not** rely on any external database, and the application's internal state should be handled by an explicit in-memory structure. The application state needs to be reset at the start of the application.

Error handling

Please assume that input parsing errors will not happen. We will not evaluate your submission against input that contains errors, is formatted incorrectly, or which breaks the contract.

Our expectations

We at Nubank value the following criteria:

- **Simplicity**: the solution is expected to be a small project and easy to understand;
- **Elegance**: the solution is expected to be easy to maintain, have a clear separation of concerns, and well-structured code organization;
- **Operational**: the solution is expected to solve the problem, cover possible corner cases, and be extensible for future design decisions.

We will look for:

- The use of referential transparency when applicable;
- Quality unit and integration tests;

- Documentation where it is needed;
- Instructions on how to run the code.

Lastly, but not least expected:

- You may use open source libraries you find suitable to support you in solving the challenge, e.g., json parsers; Please refrain as much as possible from adding frameworks and unnecessary boilerplate code.
- The challenge expects a **standalone** command-line application; please refrain from adding unnecessary infrastructure and/or dependencies. You are expected to be able to identify which tools are required to solve the problem without adding extra layers of complexity.

General notes

- This challenge may be extended by you and a Nubank engineer on a different step of the process;
- The Capital Gains application should receive the operations data through `stdin` and return the processing result through `stdout`, rather than through, for example, a REST API.

Packing The Solution for submission

- You should submit your solution source code as a compressed file (zip) containing the code and documentation. Please make sure not to include unnecessary files such as compiled binaries, libraries, etc.;
- Please do not upload your solution to public repositories such as GitHub, BitBucket, etc.

Remove Personal Information

⚠️ **IMPORTANT:** Please remove all personal information from the files of the challenge before submitting the solution. Pay special attention to the following:

- Source files like code, tests, namespaces, packaging, comments, and file names;
- Automatic comments your development environment may add to solution files;
- Code documentation such as annotations, metadata, and README.MD files;
- Version control configuration and author information.

If you plan to use git as the version control system, execute the following in the repository root to export the solution anonymized:

```
git archive --format=zip --output=./capital-gains.zip HEAD
```

Add a README

Your solution should contain a README file containing:

- Discussing the technical and architectural decisions;
- Reasoning about the frameworks used (if any framework/library was used);
- Instructions on how to compile and run the project;
- Additional notes you consider important to be evaluated.

Running Environment

It must be possible to build and run the application under Unix or Mac operating systems. Dockerized builds are welcome.