KCL Tech FirstYearHack - BlackRock Challenge

Challenge Description

As you walk into the office, you see people panicking all over! You wonder what in the world is going on. Your manager then comes to you with a frantic look and tells you that there's a hacker who has deployed a malicious trading bot on the live system and that we need to find it immediately and terminate it. However, there are also lots of good bots on the system. Fortunately, you realize that the CPU usage of the trading bots are logged. With this, you can narrow down the suspects!

Challenge #1

Malicious bots take up a lot of resources so that it lowers the available resources for good bots to use. Hence, it can slow down trades or even crash the good bots! Find the top 5 bots with highest CPU usage! It is highly likely that the malicious bots are among these 5.

Example Input:

```
1000
3000

2000
4000
6000

1000

2000
7000

5000
```

The CPU logs of each bot is differentiated by a new line. In the example log, the bot with the highest CPU usage would have a value of 12000, and the second bot would have 9000.

Task*:* Find the top 5 bots with highest CPU usage. The expected given answer will be the values of these 5 CPU usages separated by a comma (e.g. 12000, 9000, 5000, 4000, 1000).

Challenge #2

You've managed to find the top 5 bots with the highest CPU usage! Everyone in the office breathed a sigh of relief. Now, you need to single out the malicious bot from the ones you've found. You decide to look at the code of each of these 5 bots. You notice all the bots follow a simple rule-based system.

At each time step, the bots take turns looking at each security they own and trade it to another bot.

Each bot follows the following sequence:

1. It inspects its securities one by one in order (Bot Alpha checks 19 then 28)
2. It multiplies the security by its own value and then multiplies it by 0.2 to account for trading costs.
3. It then checks with its trading rule to see which bot to trade the current security to

* *In this case, trade just means give over to another bot (Bot A trades to Bot B <-> Bot A gives bot B the security)*

Trading costs are the various fees you incur when trading a security, they affect your profits, so a good bot should try to minimize them!

Example Input:

```
Bot Alpha:
  Starting securities: 19, 36
  Trading rule: If value of security divisible by 5, trade to bot Beta.
Otherwise, trade to bot Delta

Bot Beta:
  Starting securities: 15
  Trading rule: If value of security divisible by 4, trade to bot Delta.
Otherwise, trade to bot Alpha

Bot Delta:
  Starting securities: 13
  Trading rule: If value of security divisible by 8, trade to bot Alpha.
Otherwise, trade to bot Beta
---
```

Time period 1:

Bot Alpha inspects the first security with value 19. It then multiplies it by itself giving it a result of 361. It then multiplies this value by 0.2 giving the security a value of 72.2. This number is then rounded up giving a value of 73. It then checks if it's divisible by 5. Since it is not, it gives this to bot Delta. It then inspects the next item on the list, 36. It multiplies this number by itself and multiplies this value by 0.2 and rounding it up, giving a value of 260. Since this number is divisible by 5, it will give this security to bot Beta.

After bot alpha finishes its turn, the bots will have the following securities:
Bot Alpha:
Bot Beta: 15, 260

Bot Delta: 13, 73

After bot Alpha's turn, bot Beta will start inspecting and trading its securities starting from 15 then 260. Bot Delta now does the same thing.

At the end time step 1, the bots will have had inspected the following number of securities:
Bot Alpha: 2
Bot Beta: 2
Bot Delta: 3

***Task**:* Keep track of how many securities each bot inspects (including checking its initial securities) after 2 time periods. Sort the bots in descending order of total inspections.

Challenge #3

The bot with the highest number of trades seems fishy as bots should be trying to minimize costs… You are about to take the findings to senior management and show that you are a 10X developer. However, your colleague comes to you and tells you that just for today, there is no cost incurred when a trade happens and that because of this, all of the bots are trading more frequently and so you need to test for not just 2 time periods but for 5000 time periods.

This means that:

1. After multiplying the security with its own value upon inspection, it will **NOT** be multiplied by 0.2
2. You will need to simulate 5000 time periods instead of 2.

Task*:* Keep track of how many securities each bot inspects after 5000 time period. Sort the bots in descending order of total inspections and print their name.

Bonus Challenge (Optional)!

Congrats! After noticing that the malicious bot had the most trades for the whole time that it was running you were able to easily identify it!

The company was so impressed with your ability to analyse the bots behaviour, they want you to take it even further, can you come up with any more statistics about how the bots behave?

Task: Calculate any other statistics you find interesting while simulating the behaviour of the bots and give us that data.

Challenge Dataset

https://github.com/JavierMakmuri/kcl_hackathon

** Challenge 2 and 3 uses the same dataset*

Note: Because challenge 3 is well… a bit challenging, you will find the answer in the one of the files provided on GitHub (‘challenge_3_check.txt’). We will use a different dataset to the one provided when judging your answers for this challenge.

Submission Format

Please submit all your code, and the output of each challenge in a GitHub repository. You can submit the output and written details in a .txt file or in the readme. In the case you also complete the bonus challenge, please also include a document explaining what you have calculated and what makes it interesting.

Judging Criteria

For these challenges we don’t want to worry too much about performance, but we want to ensure you get the correct solution and that your code is clean, well-structured and easily understandable.

Make sure you follow good practices, like using clear variable names, consistent indentation and ensure your logic is split into multiple methods and that no one method is overly long or complex. Also, ensure your code is well commented explaining any complex logic.

In the case we can’t decide between two entries we will use the bonus challenge as a tiebreaker.