Object Oriented Programming Midterm Assessment

Introduction

In this assignment you are to create a program called advisorbot. Advisorbot is a command line program that can carry out various tasks to help a cryptocurrency investor analyse the data available on an exchange.

Example interaction with advisorbot

Advisorbot responds to commands typed in by the user.

Here is an example interaction:

```
advisorbot> Please enter a command, or help for a list of commands
user> avg ETH/BTC ask 10
advisorbot> The average ETH/BTC ask price over the last 10 timesteps was 1.0
user> predict max ETH/BTC ask
advisorbot> The max ask for ETH/BTC might be 0.9
```

From the example, you can see that the commands take several parameters.

Your job is to implement a whole set of commands in a robust way with proper error checking and implementation. The required commands are listed in the next section. It is fine to base your code off the code provided in the course. For example, you will certainly find the CSV parsing code and the tokeniser function useful. Make it clear in your source code and report which bits you wrote and which bits came from the example code or other places.

Task 1: implement commands

You advisorbot responds to a set of commands. You will need to write some sort of command parsing system which takes the command string as input and extracts the elements of the command. The extracted command elements should be checked for validity and converted into the appropriate data types. For example, the number of time frames in the avg commands should be an integer, the product name should be a string.

Here are the commands you need to implement in your code.

C1: help

Command: help

Purpose: list all available commands

Example:

```
user> help
advisorbot> The available commands are help, help <cmd>, avg, time, (etc. list all commands
```

C2: help cmd

Command: help cmd

Purpose: output help for the specified command

Example:

```
user> help avg
avg ETH/BTC bid 10 -> average ETH/BTC bid over last 10 time steps
```

C3: prod

Command: prod

Purpose: list available products

Example:

```
user> prod
advisorbot> ETH/BTC,DOGE/BTC
```

C4: min

Command: min product bid/ask

Purpose: find minimum bid or ask for product in current time step

Example:

```
user> min ETH/BTC ask
advisorbot> The min ask for ETH/BTC is 1.0
```

C5: max

Command: max product bid/ask

Purpose: find maximum bid or ask for product in current time step

Example:

```
user> max ETH/BTC ask
advisorbot> The max ask for ETH/BTC is 1.0
```

C6: avg

Command: avg product ask/bid timesteps

Purpose: compute average ask or bid for the sent product over the sent number of time steps.

Example:

```
user> avg ETH/BTC ask 10
advisorbot> The average ETH/BTC ask price over the last 10 timesteps was 1.0
```

C7: predict

Command: predict max/min product ask/bid

Purpose: predict max or min ask or bid for the sent product for the next time step

Example:

```
user> predict max ETH/BTC bid
advisorbot> The average ETH/BTC ask price over the last 10 timesteps was 1.0
```

Note that it is up to you to select an appropriate algorithm to carry out the prediction. Most people use a moving average but there are more interesting algorithms.

C8 time

Command: time

Purpose: state current time in dataset, i.e. which timeframe are we looking at

Example:

```
user> time
advisorbot> 2020/03/17 17:01:24
```

C9 step

Command: step

Purpose: move to next time step

Example:

```
user> step
advisorbot> now at 2020/03/17 17:01:30
```

Task 2: implement your own command

You should also invent your own command. It is up to you to specify what it does.

Task 3: Optimise the exchange code

You will find that the exchange code is somewhat inefficient once you start processing through it a few times. There are several points available if you are able to figure out how to make the data parsing run faster for the exchange data.

Task 4: Report, video and code

You should submit a report in PDF format and a video demonstrating your advisorbot. You should also submit your source code.

The report should contain the following items:

- An introduction describing the purpose of your advisorbot
- A table reporting which commands you were able to implement and which you were not able to implement
- A description of your command parsing code. In particular, you should explain how you validated user input and how you converted user inputs to appropriate data types to execute the commands
- A description of your custom command and how you implemented it.
- A description of how you optimised the exchange code

The code should be well commented, with comments explaining the purpose of each function.

The video should show the advisorbot running. It should demonstrate all of the commands your advisorbot can understand. The video is very important as it allows the examiner to see your program running in case they are not able to run it themselves for some reason.

Grading criteria

Your work will be assessed using the following mark scheme:

Grading item	Points
Command parsing code	5
C1: help	1
C2: help cmd	1
C3: prod	1
C4: min	2
C5: max	2
C6: avg	3
C7: predict	5
C8: time	1
C9: step	1
Task 2: Implement your own command	5
Task 3: Optimise the exchange code	5
Report	10
Overall code quality	5
Video	3
Total	50