Background

There are many stock trader software, like Chief trader, Futubull, etc, which are provided by different securities companies, banks and/or individuals. This project is to build a simple stock trader.

In this project, we consider only two types of stocks, namely, listed company and ETF.

- A stock in the type of listed company contains a stock ID, current price, latest profit per share, latest NAV per share and latest dividend.
- A stock in the type of ETF contains a stock ID, current price and a list of stock ID that the ETF includes.

A stock trader has heavy use of massive database. In this course, we use csv (comma separated values) files to simulate the database. You can assume the following files are provided under the same directory as the application (Be reminded that you should not assume the path be any absolute path):

stock-meta.csv:

Format:

StockID,Type,<list of included stock ID separated by comma>

Description:

StockID: The id of the stock, a string of five digits.

Type: The type of the stock, an integer, 1 = Listed company, 2 = ETF.

<list of included stock ID separated by comma>: This exists for ETF only. A list of stock id separated by comma. They are stocks that the ETF includes. type = 2.

daily-price.csv:

Format:

StockID,Date,Close,~~Open,High,Low,Volume~~

Description:

StockID: Same as above.

Date: The date of this record.

Close: The close price of this date, a double.

Open, High, Low and Volume are not necessary in this project. Their descriptions are omitted. In this project, we only use **Close** to be the **price**. The rest are just provided if you are interested to further extend this system after the project.

performance.csv:

Format:

StockID,Date,Profit,NAV,Dividend,PayableDate

Description:

StockId: Same as above.

Date: The date of this record.

Profit: The net profit per share of this company, a double.

NAV: The net asset value per share of this company, a double.

Dividend: The dividend per share, a double.

PayableDate: The date that the shareholders receive the dividend.

The stock trader should save the trade records and the shares that the user is holding into the following files. They should be saved in the same directory as the application. You should **not** assume the files below exist initially.

trade-record.csv:

Format:

Date,StockID,Price,#Shares,Direction

Description:

Date: The date of this trade.

StockID: Same as above.

Price: The price of this trade, a double.

#Shares: The amount of shares of this trade, an integer.

Direction: The type of this trade, an integer, 1 = Buy, 2 = Sell.

shares-holding.csv:

Format:

StockID,#Shares,AveragePrice,TotalProfit

Description:

StockID: Same as above.

#Shares: The amount of shares that the user holds, an integer.

AveragePrice: The average price for purchasing this stock, a double. See Note ^1. Note that the average price is accumulated even though the user sold all the shares and then buy it again.

TotalProfit: The total profit that the user gain (could be negative) from this stock, a double. Note that, the total profit is accumulated even though the user sold all the shares and then buy it again.

Your program should read/write these csv files for getting information in order to complete the requirements.

See the next section **Requirements** for the detail.

Requirements

(T1) Date insertion

At the beginning of the program, it asks the user to input a date as today. The date is in **DD/MM/YYYY** format.

```
Enter a date as today (DD/MM/YYYY): 28/05/2021
```

A date input is said to be **valid** if the input is

- in DD/MM/YYYY format
- a date within the date range of the daily price data file
- no earlier than the date of the last trade record

This defintion applies to the whole project description. You can assume the date input of T1 is always valid.

If the input date is not a trade date (e.g. holiday), then automatically move to the next trade date. Note that the daily price data file includes all the trade dates and it doesn't include any non-trade date.

(T2) Main Menu

Display the main menu for the user to select an action. It also displays the date of today (See example below) above the menu. Below is the options of the main menu.

1. My records
2. Enquiry a stock
3. Trade
4. Perform auto trade
5. Pass
6. Exit

If the user input is not 1, 2, 3, 4, 5 and 6, prompt the user to input again. See the example below.

```
Enter a date as today (DD/MM/YYYY): 28/05/2021
Hello! Today is 28/05/2021

1. My records
2. Enquire a stock
3. Trade
4. Perform auto Trade
5. Pass
6. Exit

Select an option: a
Invalid main menu option. Please enter again.
Select an option: 6
```

Note that the **main menu** is the **only menu** that requires validation of the option input. For **all** other sub-menus, you can assume the inputs for menu options are **valid**.

If **option 1** is selected, then see section **T3**.

If **option 2** is selected, the program prompts the user to input a stock ID. If the stock ID doesn't exists in the database, it prompts the user to input again. If the stock ID exists, then see section **T6**. Below is an

example.

```
Enter a date as today (DD/MM/YYYY): 28/05/2021
Hello! Today is 28/05/2021

1. My records
2. Enquire a stock
3. Trade
4. Perform auto Trade
5. Pass
6. Exit

Select an option: 2
Enter a stock id: 01234
Invalid stock id. Please input again.
Enter a stock id: 388
Invalid stock id. Please input again.
Enter a stock id: 0388
Invalid stock id. Please input again.
Enter a stock id:
```

If **option 3** is selected, perform the actions described in section **T7**.

If **option 4** is selected, perform the actions described in section **T8**.

If **option 5** is selected, then move to the next date and re-display T2. Note that, next date may **not** be date+1, it depends on the trade dates, e.g. holidays are excluded.

If **option 6** is selected, terminate the program.

Below is an example for option 5 and 6. Note that 29/30 May 2021 are non-trade dates.

```
Enter a date as today (DD/MM/YYYY): 28/05/2021
Hello! Today is 28/05/2021

1. My records
2. Enquire a stock
3. Trade
4. Perform auto Trade
5. Pass
6. Exit

Select an option: 5
Hello! Today is 31/05/2021

1. My records
2. Enquire a stock
3. Trade
4. Perform auto Trade
5. Pass
6. Exit

Select an option: 6
```

(T3) My records

When the user selects ***My records*** option from main menu, the following information is displayed.

- The cash remain and the overall profit (See note ^3).
- A sub-menu.
    - (*1) Each stock that the user holds at least one share is an option of this sub-menu. The option body contains
        - Stock ID
        - Total shares that the user is holding
        - Average purchase price (See note ^1)
        - Total profit from this stock (See note ^2)
    - (*2) Enquiry trading records in a period
    - Back

If an option described in (*1) above is selected, then perform the actions described in section **T5** for displaying the trade records of the selected stock.

If an option described in (*2) above is selected, then perform the actions described in section **T4**.

When the ***Back*** option is selected, the program navigates back to the previous menu. See an example below. This example first shows the content of ***My records***. The user holds some shares of 00002 and 00388, so they are displayed as an option. The option index of ****2***

becomes **3** and the option index of ***Back*** becomes **4**.

```
Cash: $1012350.30      Overall Profit: $12350.30
1. [00002]  Shares: 4000            Average Price: 73.50          Total Profit: $34215.35
2. [00388]  Shares: 1000            Average Price: 200.00         Total Profit: $53010.00
3. Enquiry trading records in period
4. Back

Select an option: 4
Hello! Today is 31/05/2021

1. My records
2. Enquire a stock
3. Trade
4. Perform auto Trade
5. Pass
6. Exit

Select an option:
```

The option index of ****2*** (Enquiry trading records in period) and ***Back*** depends on how many options for stocks. The ***stock options (*1)*** should be listed in descending order of the last trade of the stock. In the above example, the last trade of 00002 is later than the last trade of 00388 (Note, use the trade sequence instead of date because multiple stocks could be traded in the same date).

You can assume the user input of the menu options must be valid (under the assumption that your program lists the correct options).

(T4) Enquiry trading records in a period

The program prompts the user to input a start date and an end date.

You can assume each input is a valid date. However, you need to prompt the user to re-enter both start date and end date again if the end date is earlier than the start date. See an example below.

```
Cash: $1012350.30      Overall Profit: $12350.30
1. [00002]  Shares: 4000            Average Price: 73.50          Total Profit: $34215.35
2. [00388]  Shares: 1000            Average Price: 200.00         Total Profit: $53010.00
3. Enquiry trading records in period
4. Back

Select an option: 3
Enter the start date: 14/05/2021
Enter the end date: 13/05/2021
The end date cannot before the start date. Please enter again.
Enter the start date:
```

When the start and end dates are available, the program then displays the trade records (**See Section T5**) that are within the input date range.

(T5) Trade records

In this part, a **list of trade records** and a **sub-menu** are displayed. The trade records are collected according to the given information, i.e. either a **stock** (see *1 above) or a **date range** (see *2 above).

The trade records are shown line by line. Each trade record includes the date of this trade, stock ID, trade type (buy or sell), trade price, amount of shares of this trade and total trade amount. By default, the trade records are shown in the descending order of the trade sequence (Again, do not use date to determine the trade sequence).

The sub-menu includes the following options

1. Order by trade sequence (default)
2. Order by trade price
3. Order by trade shares
4. Order by total trade amount
5. Order by stock ID
6. Back

Select 1, 2, 3, 4 or 5 will re-print the trade records according to the selected order. For the tie-break (e.g. two records with the same price when ordering by price), it follows the order of trade sequence. All ordering are in descending order.

When option 6 is selected, the program navigates back to previous menu.

You can assume the input must be 1, 2, 3, 4, 5 or 6.

See an example below. It first provide a date range from T4, and then select **Order by total trade amount**, and then go back.

```
Enter the start date: 15/01/2021
Enter the end date: 02/02/2021
01/02/2021 00002 Buy  Price: 72.40          Shares: 100            Amount: $7240.00
28/01/2021 00002 Sell Price: 73.50          Shares: 200            Amount: $14700.00
18/01/2021 00002 Buy  Price: 71.75          Shares: 200            Amount: $14350.00
1. Order by trade sequence (default)
2. Order by trade price
3. Order by trade shares
4. Order by total trade amount
5. Order by stock ID
6. Back

Select an option: 4
28/01/2021 00002 Sell Price: 73.50          Shares: 200            Amount: $14700.00
18/01/2021 00002 Buy  Price: 71.75          Shares: 200            Amount: $14350.00
01/02/2021 00002 Buy  Price: 72.40          Shares: 100            Amount: $7240.00
1. Order by trade sequence (default)
2. Order by trade price
3. Order by trade shares
4. Order by total trade amount
5. Order by stock ID
6. Back

Select an option: 6
Cash: $1012350.30      Overall Profit: $12350.30
1. [00002]  Shares: 4000           Average Price: 73.50          Total Profit: $34215.35
2. [00388]  Shares: 1000           Average Price: 200.00         Total Profit: $53010.00
3. Enquiry trading records in period
4. Back

Select an option:
```

Note that dividends are **not** shown in the trade records although dividends affect the cash and profit of the user.

(T6) Enquiry a stock

Given a stock ID, it displays the following information

- Stock id and the current price (according to the current date).
- (*3) A sub-menu
- If the stock is a listed company, display the detail, namely
    - Latest net profit, net asset value and last dividend.
- (*4) If the stock is an ETF, each stock that is included in this ETF is an option of the sub-menu (*3 above). The option body is the stock ID. These stock options are displayed with following their order in the data file.
- In summary, the sub-menu (*3 above) has the following options in order
    - The stock options if it is an ETF (See *4 above)
    - Trade
    - Enquiry price within date range
    - Back

    If a **stock option** (*4 above) is selected, then recursive to ***T6 Enquiry a stock*** with this selected stock ID. Below is an example.

The user first enquiry an ETF stock (10000). The user then selects the third option (00700) and then go back.

```
[10000] Price: 29.42
1. 00003
2. 00008
3. 00700
4. Trade
5. Enquiry price within date range
6. Back

Select an option: 3
[00700] Price: 601.50          Net Profit: 20.08          NAV: 87.46          Dividend: 1.60
1. Trade
2. Enquiry price within date range
3. Back

Select an option: 3
[10000] Price: 29.42
1. 00003
2. 00008
3. 00700
4. Trade
5. Enquiry price within date range
6. Back

Select an option:
```

If the ***trade*** option is selected, then follow section **T7** with providing the corresponding stock ID.

If the ***Enquiry price within date range*** option is selected, then prompt the user to input a start date and an end date. You can assume each input is a valid date. Prompt the user to input again if the end date is earlier than the start date. When the dates are available, display the data from **start date (inclusive)** to **end date (inclusive)** line by line, each line displays the **date** and **the corresponding price**. Then, re-display T6. Note that, for price **later than today**, they **will not be shown**. If no price to be shown for a given date range, display **"no price in the date range"**. See an example below.

The example below sets 28/05/2021 be today. Note that 15,16,19,22 and 23 of May 2021 are not trade dates. There are some trade dates in between 01/06/2021 and 05/06/2021, but

no data be shown because today is set to 28/05/2021.

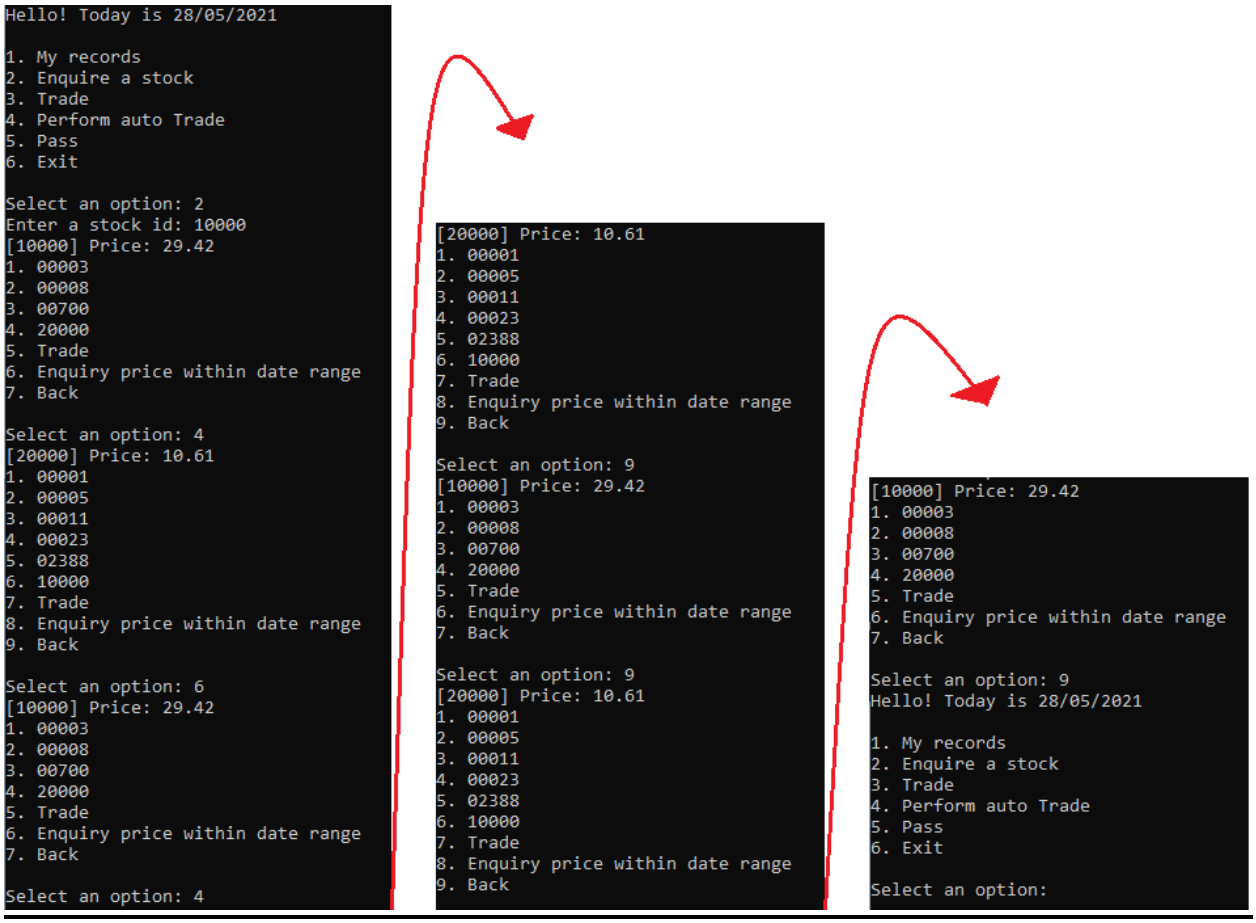
```
[00005] Price: 50.80          Net Profit: 7.56          NAV: 128.18          Dividend: 2.31
1. Trade
2. Enquiry price within date range
3. Back

Select an option: 2
Enter the start date: 14/05/2021
Enter the end date: 23/05/2021
14/05/2021: 48.75
17/05/2021: 48.55
18/05/2021: 49.20
20/05/2021: 48.45
21/05/2021: 48.55
[00005] Price: 50.80          Net Profit: 7.56          NAV: 128.18          Dividend: 2.31
1. Trade
2. Enquiry price within date range
3. Back

Select an option: 2
Enter the start date: 01/06/2021
Enter the end date: 05/06/2021
No price within the given date range.
[00005] Price: 50.80          Net Profit: 7.56          NAV: 128.18          Dividend: 2.31
1. Trade
2. Enquiry price within date range
3. Back

Select an option:
```

If ***Back*** is selected, go back to the previous menu.

You can assume the user input for menu options are valid (under the assumption that your program lists the correct options).

Note that in the given csv file **examples**, there is no ETF stock that is included in another ETF stock. But it is indeed **allowed** in this project. Suppose **ETF A includes ETF B** and **ETF B includes ETF A**. In this case, the menu *4 of ETF A shows the stock ID of ETF B. Then, if ETF B is selected, the menu jumps to show the information of ETF B which includes ETF A. Users can then select ETF A, and thus the menu may go infinitely deep, and this is **allowed**. The given csv file examples do not include this case for avoiding you get bugs from this at the beginning. You may update the examples for testing this later.

Below is an example **with adding** 10000 and 20000 be the two ETFs that are **mutually included** in the **stock-meta.csv** file.



(T7) Trade

If no stock ID is provided, then it prompts the user to input a stock ID. Same as T2 option 2, repeatedly ask for stock ID until it is an existing stock ID.

When the stock ID is known, it displays the stock ID and the current price (according to the date provided in T1) and a sub-menu. The sub-menu includes

1. Buy
2. Sell
3. Back

If option 1, ***Buy***, is selected and the cash is not enough for purchasing 1 share, then display a message “Insufficient cash” and re-display the trade menu.

If option 1, ***Buy***, is selected and the cash is enough for purchasing at least 1 share, then asks the amount of shares with displaying [1 – X] where X is the maximum amount of shares that the user can purchase (the limitation is only from the remaining cash). Prompt “Insufficient cash” and ask the user to input again if the input amount of share greater than X. You can assume the user input is an integer.

If option 2, ***Sell***, is selected and the user has no share of this stock, then display a message “Insufficient share” and re-display the trade menu.

If option 2, ***Sell***, is selected, then asks the amount of shares with displaying [1 – X] where X is the total amount of shares that the user is holding. Prompt “Insufficient share” and ask the user to input again if the user input greater than X. You can assume the user input is an integer.

If option 3, ***Back***, is selected, then go back to the previous menu.

See examples below.

Example 1: A stock id is provided from T6.

```
[00700] Price: 601.50          Net Profit: 20.08          NAV: 87.46          Dividend: 1.60
1. Trade
2. Enquiry price within date range
3. Back

Select an option: 1
[00700] Price: 601.50
1. Buy
2. Sell
3. Back

Select an option: 3
[00700] Price: 601.50          Net Profit: 20.08          NAV: 87.46          Dividend: 1.60
1. Trade
2. Enquiry price within date range
3. Back

Select an option:
```

Example 2: The stock id is not provided, so it prompts the user to enter stock id. After inputting 00700, the information and menu are shown. In this example, the user does not have enough cash for buying even 1 share and the user is not holding any share of this stock.

```
Enter a stock id: 00700
[00700] Price: 601.50
1. Buy
2. Sell
3. Back

Select an option: 1
Insufficient cash.
[00700] Price: 601.50
1. Buy
2. Sell
3. Back

Select an option: 2
Insufficient share.
[00700] Price: 601.50
1. Buy
2. Sell
3. Back

Select an option:
```

Example 3: The user has enough cash to buy the stock, but the input is greater than the maximum amount that the user can buy. Similarly, when sell is selected, the user is holding some shares of this stock, but the input is greater than the maximum amount that the user can sell.

```
Enter a stock id: 00700
[00700] Price: 601.50
1. Buy
2. Sell
3. Back

Select an option: 1
Enter the amount of shares [1-1683]: 1800
Insufficient cash.
Enter the amount of shares [1-1683]: 2
[00700] Price: 601.50
1. Buy
2. Sell
3. Back

Select an option: 2
Enter the amount of shares [1-12]: 14
Insufficient share.
Enter the amount of shares [1-12]: 4
[00700] Price: 601.50
1. Buy
2. Sell
3. Back

Select an option:
```

Once a trade is completed, the trade menu is re-displayed (See the example above). All relevant records in the files **must** be updated immediately. Also, **without** re-launch the program, all enquiries should still be most updated (i.e. including this trade), **e.g.** remaining cash, average purchase price of this stock, **etc** are all most updated.

(T8) Auto trade

It first prompts the user to input an end date. You can assume the user input is a valid date. Prompt the user to input again if the end date is earlier than ***today*** provided in T1.

When the end date is provided, the program then automatically perform trading from today until the end date.

The auto trade strategy is as following:

- The auto trade will only focus on the stock 00700
- If the current price is lower than the price of last trade date, sell all 00700 shares. If this is the first trade date (no previous trade date) or the user is not holding any 00700 shares, then do nothing.
- If the current price is higher than the price of last trade date, buy as much 00700 shares as possible. If this is the first trade date (no previous trade date) or insufficient cash for buying a 00700 share, then do nothing.

Below is an example. Today is set to 28/05/2021. The program performs auto trade from this date until 30/06/2021.

```
Enter a date as today (DD/MM/YYYY): 28/05/2021
Hello! Today is 28/05/2021

1. My records
2. Enquire a stock
3. Trade
4. Perform auto Trade
5. Pass
6. Exit

Select an option: 4
Enter the end date: 27/05/2021
The end date cannot earlier than today. Please enter again.
Enter the end date: 30/06/2021
Auto Trade: 28/05/2021 00700 Sell Price: 601.50          Shares: 10              Amount: $6015.00
Auto Trade: 31/05/2021 00700 Buy  Price: 619.00          Shares: 1635            Amount: $1012065.00
Auto Trade: 03/06/2021 00700 Sell Price: 615.00          Shares: 1635            Amount: $1005525.00
Auto Trade: 08/06/2021 00700 Buy  Price: 601.00          Shares: 1673            Amount: $1005473.00
Auto Trade: 10/06/2021 00700 Sell Price: 600.50          Shares: 1673            Amount: $1004636.50
Auto Trade: 15/06/2021 00700 Buy  Price: 600.00          Shares: 1674            Amount: $1004400.00
Auto Trade: 16/06/2021 00700 Sell Price: 596.50          Shares: 1674            Amount: $998541.00
Auto Trade: 17/06/2021 00700 Buy  Price: 599.00          Shares: 1667            Amount: $998533.00
Auto Trade: 21/06/2021 00700 Sell Price: 590.50          Shares: 1667            Amount: $984363.50
Auto Trade: 23/06/2021 00700 Buy  Price: 582.50          Shares: 1690            Amount: $984425.00
Auto Trade: 28/06/2021 00700 Sell Price: 595.50          Shares: 1690            Amount: $1006395.00
Hello! Today is 30/06/2021

1. My records
2. Enquire a stock
3. Trade
4. Perform auto Trade
5. Pass
6. Exit

Select an option:
```

The above trading result is based on the data below with the user holds 10 shares of 00700 before the auto trade.

```
00700,27/05/2021,605.500000,
00700,28/05/2021,601.500000,
00700,31/05/2021,619.000000,
00700,01/06/2021,628.000000,
00700,02/06/2021,628.500000,
00700,03/06/2021,615.500000,
00700,04/06/2021,611.500000,
00700,07/06/2021,600.500000,
00700,08/06/2021,601.000000,
00700,09/06/2021,603.000000,
00700,10/06/2021,600.500000,
00700,11/06/2021,596.000000,
00700,15/06/2021,600.000000,
00700,16/06/2021,596.500000,
00700,17/06/2021,599.000000,
00700,18/06/2021,603.000000,
00700,21/06/2021,590.500000,
00700,22/06/2021,573.500000,
00700,23/06/2021,582.500000,
00700,24/06/2021,583.000000,
00700,25/06/2021,598.500000,
00700,28/06/2021,595.500000,
00700,29/06/2021,590.500000,
00700,30/06/2021,584.000000,
```

On 28/05/2021, the program sells all the 10 shares of 00700 because 601.5 < 605.5.

On 31/05/2021, the program buys as much 00700 shares as possible because 619 > 601.5

On 03/06/2021, the program sells all 00700 shares because 615.5 < 628.5

On 08/06/2021, the program buys as much 00700 shares as possible because 601 > 600.5

On 10/06/2021, the program sells all 00700 shares because 600.5 < 603.

On 15/06/2021, the program buys as much 00700 shares as possible because 600 > 596.

On 16/06/2021, the program sells all 00700 shares because 596.5 < 600.

On 17/06/2021, the program buys as much 00700 shares as possible because 599 > 596.5.

On 21/06/2021, the program sells all 00700 shares because 590.5 < 603.

On 23/06/2021, the program buys as much 00700 shares as possible because 582.5 > 573.5.

On 28/06/2021, the program sells all 00700 shares because 595.5 < 598.5.

When the auto trade is completed, the program stays on the specified end date, 30/06/2021 of the above example.

Out of this project:

Of course, this auto trade strategy cannot earn, but just for course project purpose. The given data provides extra information, the open, high, low, volume. After the course project, if you are interested, you may implement your own strategies and see whether it earns. For example, you may consider computing some technical analysis indicators like MACD, RSI, etc and/or detecting some patterns like Hammer, Flag, Ascending triangle, etc.

Notes

Note ^1: The definition of average purchase price may be different with the real-world one for simplicity. It is simply (p_1*s_1 + p_2*s_2 + … + p_n*s_n)/(s_1 + s_2 + … + s_n) where p_i is the price of the i-th purchase of this stock and s_i is the amount of share of the i-th purchase of this stock with IGNORING all the sell actions. It counts ALL trades from the beginning of the use of the system (not only the current execution, i.e. do some trades, then terminate the system, and then relaunch the system. All records should still be counted).

Note ^2: Total profit of a stock includes the dividend. At the date of receiving the dividend of stock X, if you have Y shares of X and the dividend is $Z per share, then you earn $Z*Y. Also, the total profit of a stock includes all the past trading of this stock from the beginning of the use of this system (not only the current execution, i.e. do some trades, then terminate the system, and then relaunch the system. All records should still be counted)..

Note ^3: Overall profit include the profit over all stocks even though the user is not currently holding any share of the stock (i.e. earned before, but all sold now, they are also counted in overall profit).

Note ^4: The user may not launch the system every day. Your program should update the user information when launch and/or perform some operations. E.g. the user holds some shares of 00700 on 03/01/2020 and this user does not launch the system until 28/05/2021. The user should receive the dividends of 00700 from 03/01/2020 to 28/05/2021. When the user launch the system, the cash and overall profit should be updated because some dividends are received.

Again, most of the definitions in this project are simplified from the real situation for course project purpose.

Your submission should not contain any Java statements for printing to screen except calling the given class (Screen) methods for printing.

Your submission should not contain any objects for reading keyboard input except using the Scanner object, keyboard, provided in the given class (Screen).