Take Home Task Description

- This take home task consists of two parts. Please finish the first part before moving on to the second part; there will be a marking in this document to inform you where the first task ends.
- Once you have finished this task, you may proceed to attempt the question sent to you on HackerRank. **Please only open the HackerRank once you have completed this take home task.**
- We suggest that you keep your solution to one file. Feel free to use comments to break up your code into sections or regions as if you were importing them.
- The use of version control is not required (however, if you choose to use VCS, please keep the repo private).
- The suggested time to complete both parts is two and a half hours. However, please feel free to spend more or less time on this take home task as you see fit.
- Please use this opportunity to demonstrate your proper software engineering skills.
- Have fun and good luck!

Note: Please finish this take home task in one of the following languages:
C++14, C#, Java8, Python3, Scala

Part 1

Introduction

An "**Order Matching Engine**", or simply "**Matching Engine**", is an electronic system that matches "**Buy**" and "**Sell**" orders for a stock market. A trader can submit several types of orders to a matching engine, which will process the submitted orders and possibly place them into an **Order Book (OB)**, which is an electronic list of buy and sell orders organised by price level.

Your task for this take home task is to design and implement a Matching Engine which maintains an OB. This Matching Engine should support multiple different types of submitted orders, as well a few other possible actions that a Trader might want to perform.

Order Book (OB) Description

As mentioned in the introduction, the OB is a list of buy and sell orders. Specifically, it contains two lists: a **list of Buy orders** and a **list of Sell orders**, sorted in the order of decreasing priority (i.e. the order with the highest priority is at the front of the list).

- For Buy orders, a Buy order with **a higher Price** has higher priority than one with a lower Price. If two Buy orders have the same Price, the order that arrived in the OB **earlier** will have higher priority.
- For Sell orders, a Sell order with **a lower Price** has higher priority than one with a higher Price. If two Sell orders have the same Price, the order that arrived in the OB **earlier** will have higher priority.

Each order will have the following attributes:

- **Side**: Either 'B' or 'S', representing the 'Buy' and 'Sell' sides respectively.
- **Order ID (string)**: A unique identifier consisting of exactly 4 alphanumeric characters. This ID is case-sensitive (i.e. "abcd" is different from "ABcd").
- **Price (integer)**: The best price at which the trader is willing to buy or sell a single unit of stock. For a Buy order, this represents the maximum price at which they are willing to buy. For a Sell order, this represents the minimum price at which they are willing to sell.
- **Quantity (integer)**: The number of units of stock that the trader is willing to buy or sell.

*Note that certain Order Types may have additional attributes, or only a subset of these attributes - more details can be found in the **Types of Orders** section.*

The orders within the OB are represented in the form **'[Quantity]@[Price]#[Order ID]'**. For example, a Buy order represented as '**450@7#iv15**' represents a Buy order with the Order ID **'iv15'**, where the trader is willing to buy 450 units of stock at a price of 7 per unit of stock.

The following is an example of an OB:

```
B: ['300@9#ey7r', '450@7#iv15']
S: ['250@12#bG0d', '250@12#Ljup']
```

Where **B** represents the list of Buy orders, and **S** represents the list of Sell orders, sorted in descending order of priority. In this example, order **'ey7r'** is the Buy order with the highest priority.

Order Matching/Execution

When an order gets submitted to the OB, it will get executed (sometimes partially) if there is a **match** with an order in the opposing Side within the OB (That is, a submitted Buy order will be matched with the Sell orders within the OB, and vice versa). A match occurs between a Buy order and a Sell order if the **Price of the Buy order is greater than, or equal to, the Price of the Sell order**. The quantity matched (executed) is equal to the lower of the quantities of the two matched orders, and the price used for the trade will be the **Price** of the Order in the OB (and not the submitted order).

The matching engine will repeatedly attempt to match the submitted order with the order of highest priority in the opposing side within the OB and execute trades upon successful matches.

Consider the following OB, where a Trader has just submitted a Buy order with ID **'QpoZ'** for 500 units of stock at a price of 12 per unit of stock:

```
Submitted buy order: '500@12#QpoZ'

B: ['300@9#ey7r', '450@7#iv15']
S: ['280@10#z14R', '200@12#bG0d', '250@14#Ljup']
```

A match occurs between Buy order **'QpoZ'** (with Price 12) and Sell order **'z14R'** (with Price 10), where the quantity matched is 280 (the lower of 500 and 280). Thus, a trade will occur between orders **'QpoZ'** and **'z14R'** for 280 units of stock at a price of 10 per unit of stock.

Since the Sell order **'z14R'** is fully executed (i.e. its remaining quantity is reduced to zero), it will be removed from the OB. The Buy order **'QpoZ'** still has a remaining quantity of 220. The OB now looks like this:

```
Submitted buy order: '220@12#QpoZ'

B: ['300@9#ey7r', '450@7#iv15']
S: ['200@12#bG0d', '250@14#Ljup']
```

A match occurs between Buy order '**QpoZ**' and Sell order '**bG0d**', and so the same process will occur. The Sell order '**bG0d**' is fully executed and will be removed from the OB. The Buy order '**QpoZ**' has a remaining quantity of 20.

```
Submitted buy order: '20@12#QpoZ'

B: ['300@9#ey7r', '450@7#iv15']
S: ['250@14#Ljup']
```

At this point, the Buy order 'QpoZ' cannot be matched with any remaining Sell order. It will be added to the OB with its remaining quantity.

```
B: ['20@12#QpoZ', '300@9#ey7r', '450@7#iv15']
S: ['250@14#Ljup']
```

A total of 480 units of stock was traded during the execution of this order, and the total cost is 280 * 10 + 200 * 12 = 5200.

*Note that this is procedure for the execution of a **Limit Order**; other Order types may be executed differently, as seen in the **Types of Orders** section.*

Types of Orders

These are the possible Orders that a Trader can submit to the Matching Engine.

Limit Order

All of the Orders that have been discussed in previous sections are Limit Orders. A limit order consists of a **Side** (either '**B**' or '**S**', for the 'Buy' and 'Sell' sides respectively), a **Quantity**, and a **Price.**

Upon submitting a Limit Order, the Matching Engine will execute the submitted order as long as there is a match with the opposing Side within the OB. If the limit order is not executed fully, it will be added to the OB with its remaining quantity, as described in the **Order Matching/Execution** section.

Market Order

Unlike the Limit Order, a Market Order only consists of a **Side** and a **Quantity**; it **does not have a Price**.

Upon submitting a Market Order, the Matching Engine will repeatedly match the submitted order with the order of highest priority in the opposing side within the OB, regardless of Price. If the market order is not executed fully, the remaining quantity will be cancelled.

For example, suppose the OB currently looks like this:

```
B: ['300@9#ey7r', '450@7#iv15']
S: ['250@14#Ljup', '290@16#pQ34']
```

If a **Sell** Market Order is submitted with a **Quantity** of 400, it will be matched with order **'ey7r'** for 300 units of stock, then with order **'iv15'** for 100 units of stock. The OB now looks like this:

```
B: ['350@7#iv15']
S: ['250@14#Ljup', '290@16#pQ34']
```

If another **Sell** Market Order is submitted with a **Quantity** of 500, it will be matched with order **'iv15'** for 350 units of stock.

```
B: []
S: ['250@14#Ljup', '290@16#pQ34']
```

Since there are no more Buy orders in the OB, no matches can be made. The remaining quantity of 150 in the Market Order will be cancelled.

Other Actions

Aside from submitting orders, there is another action that a Trader can perform.

Cancel Order

Cancel an Order, hence removing it from the OB.

Grading Criteria

Your solution will be manually reviewed and evaluated based on **correctness**, **design** and **efficiency.** Furthermore, good software engineering practices should ideally be adhered to.

Input Format

You may assume that the OB initially starts out empty.

The input consists of a sequence of commands - one command per line of input. Each line of input will be one of the following:

SUB LO [Side] [Order ID] [Quantity] [Price]
Submit a Limit Order with the specified Order ID, Quantity and Price.

SUB MO [Side] [Order ID] [Quantity]
Submit a Market Order with the specified Order ID and Quantity.

CXL [Order ID]
Cancel the order in the OB with the specified Order ID. If there is no order in the OB with the specified Order ID, this action should do nothing.

The input will terminate with a single line, **END**, representing the end of the sequence of actions.

Output Format

For each action that represents a submitted order (i.e. any action beginning with 'SUB'), output a single integer: the total cost of the units of stock that was traded during the execution of the order. If no stocks were traded during the execution, output 0.

Finally, at the end of the sequence of actions, print out the entire OB in two lines. The first line should be the list of Buy Orders in the OB; the second line should be the list of Sell Orders in the OB. The list of Buy orders should start with the string **'B: '** (without the quotes), followed by the orders in decreasing priority, separated by spaces. The list of sell orders should be printed in the same way, but beginning with **'S: '** instead of 'B: '. Each order should be printed in the format **'[Quantity]@[Price]#[Order ID]'.**

Sample Input

```
SUB LO B Ffuj 200 13
SUB LO B Yy7P 150 11
SUB LO B YuFU 100 13
SUB LO S IpD8 150 14
SUB LO S y93N 190 15
SUB LO B Y5wb 230 14
SUB MO B IZLO 250
CXL Ffuj
CXL 49Ze
END
```

Sample Output

```
0
0
0
0
0
2100
2850
B: 80@14#Y5wb 100@13#YuFU 150@11#Yy7P
S:
```

Explanation

The first 5 orders are Limit Orders that do not get executed; they simply get added to the Order Book, which now looks like this:

```
B: 200@13#Ffuj 100@13#YuFU 150@11#Yy7P
S: 150@14#IpD8 190@15#y93N
```

The 6th order, '**Y5wb**' (Buy Limit Order), matches with order '**IpD8**' for 150 units of stock at a price of 14 per unit. The total cost is 150 * 14 = 2100. The remaining 80 units is added to the list of Buy orders.

The 7th order, '**IZLO**' (Buy Market Order), matches with order '**y93N**' for 190 units of stock at a price of 15 per unit. The total cost is 190 * 15 = 2850. The remaining 60 units is cancelled.

```
B: 80@14#Y5wb 200@13#Ffuj 100@13#YuFU 150@11#Yy7P
S:
```

The 8th command cancels order '**Ffuj**'.

The 9th command attempts to cancel order '**49Ze**', but no such order exists in the OB. Therefore, nothing will happen.

Constraints

Each test case contains at most 100 commands.

All numerical values (i.e. **Quantities**, **Prices**) are positive integers. You are guaranteed that any integer that appear in the input, as well as any integer value to be printed in the output, will fit in a signed 32-bit integer.

--- PART 1 ENDS HERE ---

Part 2

Introduction

Part 2 of the Matching Engine introduces a few new Order Types and Actions.

New Order Types

Immediate-or-Cancel (IOC) Order

An IOC Order is similar to a Limit Order, except if the IOC Order is not executed fully, the remaining quantity will be cancelled, instead of being added to the OB. Like the Limit Order, an IOC Order consists of a **Side**, a **Quantity** and a **Price**.

For example, suppose the OB currently looks like this:

```
B: ['300@9#ey7r', '450@7#iv15']
S: ['250@14#Ljup', '290@16#pQ34']
```

If a **Sell** IOC Order is submitted with a **Quantity** of 400 and a **Price** of 8, it will be matched with order '**ey7r**' for 300 units of stock. The OB now looks like this:

```
B: ['450@7#iv15']
S: ['250@14#Ljup', '290@16#pQ34']
```

The IOC Order can no longer be matched with any Buy order. The remaining quantity of 100 is cancelled.

Fill-or-Kill (FOK) Order

An FOK Order is similar to a Limit Order, except the FOK Order will be executed **if and only if it can be executed fully**. If the FOK Order **cannot** be executed fully, it will **not be executed at all** - that is, the OB will **remain exactly the same**.

For example, suppose the OB currently looks like this

```
B: ['300@9#ey7r', '450@7#iv15']
S: ['250@14#Ljup', '290@16#pQ34']
```

If a **Sell** FOK Order is submitted with a **Quantity** of 200 and a **Price** of 8, it can be executed fully by being matched with order 'ey7r' for 200 units of stock. The OB now looks like this

```
B: ['100@9#ey7r', '450@7#iv15']
S: ['250@14#Ljup', '290@16#pQ34']
```

If another **Sell** FOK Order is submitted with a **Quantity** of 200 and a **Price** of 8, it can only be executed partially. Therefore, the entire FOK order is cancelled and the OB will remain exactly the same.

New Actions

Cancel & Replace Limit Order

Cancels and replaces a Limit Order in the OB with a new **Price** and **Quantity**, keeping the order **ID** the same. Effectively serves as an update to the **Quantity** and/or **Price** of a Limit Order within the OB.

The updated order will be treated as a newly-inserted order, which means it will be given lower priority compared to other existing orders of the same **Price**. The only exception to this rule is when the **Price remains the same** and the **Quantity decreases** (or also remains the same), in which case, the order's priority will remain the same.

For example, suppose the OB currently looks like this:

```
B: ['300@9#ey7r', '450@7#iv15']
S: ['250@14#Ljup', '290@14#pQ34', '200@14#o3DS', '130@16#xb1X']
```

If order '**pQ34**' has its **Quantity** updated from 290 to 190 (and its **Price** remain the same), its priority will remain the same.

```
B: ['300@9#ey7r', '450@7#iv15']
S: ['250@14#Ljup', '190@14#pQ34', '200@14#o3DS', '130@16#xb1X']
```

However, if the same order has its **Quantity** now updated from 190 to 500 (and its **Price** remain the same), it will be treated as a new order, causing it to have a lower priority than the order '**o3DS**'.

```
B: ['300@9#ey7r', '450@7#iv15']
S: ['250@14#Ljup', '200@14#o3DS', '500@14#pQ34', '130@16#xblX']
```

You are guaranteed that an updated price will not cause any matches to occur. (For example, a request to update the price of order '**ey7r**' to 15 will not occur, since that will match with order '**Ljup**')

Input Format

The input format of the new commands are as follows:

SUB IOC [Side] [Order ID] [Quantity] [Price]
Submit an Immediate-Or-Cancel (IOC) Order with the specified Order ID, Quantity and Price.

SUB FOK [Side] [Order ID] [Quantity] [Price]
Submit a Fill-Or-Kill (FOK) Order with the specified Order ID, Quantity and Price.

CRP [Order ID] [New Quantity] [New Price]
Update the Quantity and Price of the Limit Order with the specified Order ID. If the specified Order ID does not correspond to a **Limit Order** in the OB, this action should do nothing.

Note that commands that appear in previous parts (e.g. **SUB LO**) may appear in these test cases as well.

Output Format

The output format is exactly the same as Part 1.

Sample Input

```
SUB LO B N1Eh 300 12
SUB LO B 0Gxb 250 11
SUB LO S JSvU 350 14
SUB LO S uH6w 320 15
SUB IOC S ckMR 150 10
SUB IOC B DVeP 500 14
SUB FOK S ejnR 200 12
SUB FOK S 8uGs 200 9
SUB LO B 2va9 250 12
SUB LO B 9zS1 300 11
CRP 2va9 480 11
CRP 9zS1 170 11
END
```

Sample Output

```
0
0
0
0
1800
4900
0
2350
0
0
B: 200@11#0Gxb 170@11#9zS1 480@11#2va9
S: 320@15#uH6w
```

Explanation

After the first 4 Limit Orders, the OB looks like this:

```
B: 300@12#N1Eh 250@11#0Gxb
S: 350@14#JSvU 320@15#uH6w
```

The 5th order, '**ckMR**' (Sell IOC Order), matches with order '**N1Eh**' for 150 units of stock at a price of 12 per unit. The total cost is 150 * 12 = 1800; the IOC order is fully executed.

The 6th order, '**DVeP**s' (Buy IOC Order), matches with order 'JSvU' for 350 units of stock at a price of 14 per unit. The total cost is 350 * 14 = 4900. The remaining 150 units of the order is cancelled.

```
B: 150@12#N1Eh 250@11#0Gxb
S: 320@15#uH6w
```

For the 7th order, '**ejnR**' (Sell FOK Order), attempting to execute it will only cause it to be matched for 150 units of stock (out of 200). Since it cannot be executed fully, it will not be executed.

For the 8th order, '**8uGs**' (Sell FOK Order), it is able to fully execute by matching with '**N1Eh**' for 150 units of stock and '**0Gxb**' for 50 units of stock. The total cost is 150 * 12 + 50 * 11 = 2350.

The 9th and 10th Limit Orders will be added to the OB.

```
B: 250@12#2va9 200@11#0Gxb 300@11#9zS1
S: 320@15#uH6w
```

The 11th command will update order '**2va9**' from 250 to 480 for Quantity, and from 12 to 11 for Price. It will have the lowest priority among Buy orders with Price 11.

```
B: 200@11#0Gxb 300@11#9zS1 480@11#2va9
S: 320@15#uH6w
```

The 12th command will update order '**9zS1**' from 300 to 170 for Quantity. Since Price did not change and Quantity decreased, its priority will remain the same.

```
B: 200@11#0Gxb 170@11#9zS1 480@11#2va9
S: 320@15#uH6w
```

--- PART 2 ENDS HERE ---

When you are ready, please attempt the HackerRank problem by following the link in the email. Thank you!