The task is to implement an order book. Exchange matching engines use an order book to match orders from buyers and sellers. It’s a simple mechanism which can be best described using small examples.

Let’s consider order flow for some imaginary company traded under symbol XYZ. Initially, an order book has no orders:

==========

Sellers (ASK)

Buyers (BID)

==========

Let’s say some (anonymous) market participant places a sell order for 5 shares at a price of 110 USD. Then order book will look like:

==========

ASK

110: 5

BID

==========

A moment after that some other market participant places a buy order for 10 shares at a price of 90 USD:

==========

ASK

110: 5

90: 10

BID

==========

So far we've not had any trades because the lowest price from the seller (best ASK) is 110 USD. And the highest price from the buyer (best BID) is 90 USD. In other words, currently there is no match between buyers and sellers. A difference between best BID and best ASK is called the spread and in this case equals to 20 USD.

A few moments later, a new sell order has been added - sell 10 shares at 110 USD:

==========

ASK

110: 5 10

90: 10

BID

==========

Note that a new order at the price level 110 has been added to the end of the order queue. It means that in case of match, the previous order (5 shares) will be executed before newly added order.

Let's say a few moments later, several more orders from buyers and sellers are added to our order book, but no trades happened yet.

==========

ASK

110: 5 10

105: 3 7

100: 4 6

90: 10 2 3

BID

==========

Now, let’s imagine some buyer places an “aggressive” order to buy 4 shares at the price of 105 USD. It will be an order which actually matches the best price from sellers (namely lowest price at 105 USD). The seller’s order of 3 shares at price 105 USD was added earlier than another seller’s order of 7 shares at 105 USD. Therefore seller’s order of 3 shares will be matched first.

As a result we will see a trade:

3 shares of XYZ were sold at 105 USD

And the order book will look like:

==========

ASK

110: 5 10

105: 7

100: 4 6

90: 10 2 3

BID

==========

But we still have 1 share left from an “aggressive” buyer who wants to buy at 105 USD. Therefore we will have partial match with seller’s order at 105 USD. We will see second trade happening:

1 share of XYZ were sold at 105 USD

And now the order book will now look like:

==========

ASK

110: 5 10

105: 6

100: 4 6

90: 10 2 3

BID

==========

In order words, that order from the “aggressive” buyer crossed the book and two trades were generated.

Now let’s imagine that some other “aggressive” seller wants to sell 23 shares at 80 USD. In other words, he or she wants to sell 23 shares no cheaper than 80 USD.

First trade (or execution in other terminology) will be 4 shares at price of 100 USD because buy order of 4 shares at price level of 100 was added before buy order of 6 shares at price level 100. The order book will look like:

==========

ASK

110: 5 10

105: 6

100: 6

90: 10 2 3

BID

==========

Second trade will be 6 shares at price 100 USD:

==========

ASK

110: 5 10

105: 6

90: 10 2 3

BID

==========

Third trade will be 10 shares at price 90 USD:

==========

ASK

110: 5 10

105: 6

90: 2 3

BID

==========

Fourth trade will be 2 shares at price 90 USD:

==========

ASK

110: 5 10

105: 6

90: 3

BID

==========

And finally the last trade will be 1 share at price 90 USD (partial match or partial execution):

==========

ASK

110: 5 10

105: 6

90: 2

BID

==========

Now let’s say we have got a new buy order of 8 shares at the price 107 USD.

We will see trade:

6 shares of XYZ were sold at 105 USD:

==========

ASK

110: 5 10

90: 2

BID

==========

We still have 2 more shares from a buyer who is willing to buy no higher than 107 USD but the best sell order right now is at 110 USD so we place a new order at the level of 107 USD and the order book will look like:

==========

ASK

110: 5 10

107: 2

90: 2

BID

==========

As you might noticed the first priority is price, the second priority is time when order was placed at certain price level. It's called price/time limit order book. Usually in trading interfaces you might see only a sum of all orders at each price level and only few price levels around best BID/ASK price.

There are also delete orders if a trader wants to withdraw his or her order from the market (obviously each order has unique order id). For obvious reasons if trader deletes order and then adds order, new order will be added to the end of the queue.

Your implementation of the order book should read market data file and build order book, print to stdout trades which happened and final state of order book.

Market data file format:

Each line represents order added in chronological order.

First byte of each line represents message type:

- 'A' - new order
- 'X' - delete order

Order has format for rest of a line: order_id, side, quantity, price

For example:

A,16113575,B,18,585

It decoded as:

A = add new order

16113575 = order id

B = buy order

18 = number of shares to buy

585 = price

Example of input file:

A,100000,S,1,1075

A,100001,B,9,1000

A,100002,B,30,975

A,100003,S,10,1050

A,100004,B,10,950

A,100005,S,2,1025

A,100006,B,1,1000

X,100004,B,10,950

A,100007,S,5,1025

A,100008,B,3,1050

X,100008,B,3,1050

X,100005,S,2,1025

Bonus points: testability and performance notes.