Problem I. Investment Investigation

Source file name:	Investment.c, Investment.cpp, Investment.java, Investment.py
Input:	`Standard`
Output:	`Standard`

To make some extra money on the side, you have recently started running your own cryptocurrency exchange, where people can trade their Budget Amplifying Profit Coin (BAPC). It is quickly gaining popularity, however, this has also resulted in government regulators asking some questions… As part of their investigation, they have asked for a list of all transactions that have been made via your exchange. You have never bothered to keep track of this, but luckily, you still have the list of all orders that were made since the start of the exchange.

Contentedly looking at the value of your BAPC going through the roof. Internet meme, fair use

The exchange operates by keeping a list of outstanding buy and sell orders, each with a price and an amount. Whenever a *normal* order comes in, it is checked whether the new lowest sell price is less than or equal to the highest buy price. If this is the case, a transaction is made between the sell order with the lowest price and the buy order with the highest price, such that at least one of these orders is completely fulfilled. In case of a tie in price, older orders are fulfilled first. This is repeated until the lowest sell price is strictly larger than the highest buy price.

If instead a *Fill-or-Kill* (FoK) buy order comes in, there must currently be enough outstanding sell orders with a price of at most the offered price to completely fulfil this order. If there are, the order will be fulfilled in the same way as a normal order. Otherwise, the order is completely cancelled, without any transaction taking place. Note that multiple orders may be used to complete a FoK order, as long as it happens immediately.

FoK sell orders are processed in a similar way, but then there should be sufficient outstanding buy orders with a price of at least the asked price.

As an example, consider the first sample case. The six orders are handled as follows:

1. The first order is added to the list of outstanding orders.
2. The second order is partially fulfilled by selling 10 BAPC to the first order. This removes the first order from the list of outstanding orders, and adds the remainder of the second order (consisting of 10 BAPC) to this list.
3. The third order is added to the list of outstanding orders.
4. The fourth order is a FoK buy order that cannot be immediately fulfilled, so it is ignored. It is not added to the list of outstanding orders.
5. The fifth order can be immediately fulfilled by first buying 10 BAPC from order 2 and then buying 50 BAPC from order 3. The resulting list of outstanding orders only consists of the remaining 8 BAPC of order 3.
6. The sixth order is added to the list of outstanding orders.

Given a list of all orders in the order that they have been made, create a list of all transactions that have been performed by your exchange.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^5$), the number of orders.

- n lines, each describing an order:
  - A string s, either "`buy`" or "`sell`", the side of the order.
  - A string t, either "`normal`" or "`fok`", the type of the order.
  - An integer p ($1 \leq p \leq 10^9$), the offered or asked price per BAPC.
  - An integer a ($1 \leq a \leq 10^9$), the amount of BAPC being asked or offered.

Output

The output consists of the number of performed transactions, and then for each transaction, in the order that they have been performed:

- The index of the corresponding "`sell`" order.
- The index of the corresponding "`buy`" order.
- The amount of BAPC being traded.

Here, the index of an order is its position in the input, where the first order has index 1.

Example

Input	Output

6
buy normal 700 10
sell normal 500 20
sell normal 800 58
buy fok 600 30
buy fok 900 60
sell normal 300 42
	3
	2 1 10
	2 5 10
	3 5 50

3
buy normal 19 10
buy normal 19 20
sell fok 19 17
	2
	3 1 10
	3 2 7