1598 Exchange

You are taking part in a large project to automate operations for Northeastern Exchange of Resources and Commodities (NEERC). Different resources and commodities are traded on this exchange via public auction. Each resource or commodity is traded independently of the others and your task is to write a core engine for this exchange — its order book. There is a separate instance of an order book for each traded resource or commodity and it is not your problem to get the correct orders into order books. The order book instance you will be writing is going to receive the appropriate orders from the rest of exchange system.

Order book receives a stream of *messages*. Messages are *orders* and requests to cancel previously issued orders. Orders that were not cancelled are called *active*. There are orders to *buy* and orders to *sell*. Each order to buy or to sell has a positive *size* and a positive *price*. Order book maintains a list of active orders and generates *quotes* and *trades*. Active order to buy at the highest price is the best buy order and its price is called *bid* price. Active order to sell at the lowest price is the best sell order and its price is called *ask* price. Ask price is always lower than bid price, that is, buyers are willing to pay less than sellers want to receive in return.

A current quote from the order book contains current bid size, bid price, ask size, and ask price. Here bid and ask sizes are sums of the the sizes of all active orders with the current bid price and the current ask price correspondingly.

A trade records information about transaction between buyer and seller. Each trade has size and price.

If an order to buy arrives to the order book at a price greater or equal to the current ask price, then the corresponding orders are *matched* and trade happens — buyer and seller reached agreement on a price. Vice versa, if an order to sell arrives to the order book at a price less or equal to the current bid price, then trade happens, too. For the purpose of order matching, order book works like a FIFO queue for orders with the same price (read further for details).

When an order to buy arrives to the order book at a price greater or equal to the current ask price it is not immediately entered into the order book. First, a number of trades is generated, possibly reducing the size of incoming order. Trade is generated between incoming buy order and the best order to sell. If there are multiple best orders (at the ask price), then the order that entered the order book first is chosen. Trade is generated at the current ask price with the size of the trade being equal to the smaller of the sizes of two matching orders. Sizes of both matching orders are reduced by the size of the trade. If that reduces the size of sell order to zero, then it becomes inactive and is removed from the order book. If the size of incoming buy order becomes zero, then the process is over — incoming order becomes inactive. If the size of incoming buy order is still positive and there is another sell order to match with, then the process continues generating further trades at the new ask price (ask price can increase as sell orders are traded against and become inactive). If there is no sell order to match with (current ask price became greater than incoming buy order price), then incoming buy order is added to the order book with its remaining size.

For incoming sell order everything works similarly — it is matched with buy orders from the order book and trades are generated on bid price.

On incoming cancel request the corresponding order is simply removed from the order book and becomes inactive. Note, that by the time of the cancel request the quantity of the corresponding order might have been already partially reduced or the order might have become inactive. Requests to cancel inactive order do not change anything in the order book.

On every incoming message the order book has to generate all trades it causes and the current quote (bid size, bid price, ask size, ask price) after processing of the corresponding message, even when

nothing has changed in the order book as a result of this message. Thus, the number of quotes the order book generates is always equal to the number of incoming messages.

Input

The input will contain several test cases, each of them as described below. Consecutive test cases are separated by a single blank line.

The first line of the input contains a single integer number n ($1 \leq n \leq 10000$) — the number of incoming messages that the order book has to process. The following n lines contain messages. Each line starts with a word describing the message type — '`BUY`', '`SELL`', or '`CANCEL`' followed after a space by the message parameters.

'`BUY`' and '`SELL`' denote an order to buy or to sell correspondingly, and are followed by two integers q and p ($1 \leq q \leq 99999$, $1 \leq p \leq 99999$) — order size and price. '`CANCEL`' denotes a request to cancel previously issued order. It is followed by a single integer i which is the number of the message with some preceding order to buy or to sell (messages are numbered from 1 to n).

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Write to the output a stream of quotes and trades that the incoming messages generate. For every trade write '`TRADE`' followed after space by the trade size and price. For every quote write '`QUOTE`' followed after space by the quote bid size, bid price, minus sign ('`-`'), ask size, ask price (all separated by spaces).

There is a special case when there are no active orders to buy or to sell in the order book (bid and/or ask are not defined). This case is treated as follows. If there is no active order to buy, then it is assumed that bid size is zero and bid price is zero. If there is no active order to sell, then it is assumed that ask size is zero and ask price is 99 999. Note, that zero is not a legal price, but 99 999 is a legal price. Recipient of quote messages distinguishes actual 99 999 ask price from the special case of absent orders to sell by looking at its ask size.

See example for further clarification.

Sample Input

```
11
BUY 100 35
CANCEL 1
BUY 100 34
SELL 150 36
SELL 300 37
SELL 100 36
BUY 100 38
CANCEL 4
CANCEL 7
BUY 200 32
SELL 500 30
```

Sample Output

```
QUOTE 100 35 - 0 99999
QUOTE 0 0 - 0 99999
QUOTE 100 34 - 0 99999
```

```
QUOTE 100 34 - 150 36
QUOTE 100 34 - 150 36
QUOTE 100 34 - 250 36
TRADE 100 36
QUOTE 100 34 - 150 36
QUOTE 100 34 - 100 36
QUOTE 100 34 - 100 36
QUOTE 100 34 - 100 36
TRADE 100 34
TRADE 200 32
QUOTE 0 0 - 200 30
```