Book Analyzer

Background

Suppose you unexpectedly inherit $3000 which you decide you want to invest in the Zany Interesting Nifty Gizmo company (stock symbol: ZING). You are willing to pay up to $30 per share of ZING. So you log in to your online trading account and enter a limit order: "BUY 100 ZING @ $30". It's a limit order because that's most you're willing to pay. You'd be willing to pay less than $30, but not more than $30. Your order will sit around in the market until you get your 100 shares. A limit order to buy is called a "bid".

But you're not the only prospective buyer for ZING stock. Others want to buy it too. One is bidding $31/share for 200 shares, while another is bidding $29/share for 300 shares. When Mark Cuban wants to sell 225 shares, he's obviously going to take the highest price he can get for each share. So he hits the $31 bid first, selling 200 shares. Then he sells his remaining 25 shares to you at $30/share. Your bid size reduced by 25, leaving 75 shares still to be bought.

Suppose you eventually get the full 100 shares at some price. Next year, you decide to buy a new computer and you need $4500 for it, and luckily the value of ZING has appreciated by 50%. So you want to sell your 100 shares of ZING stock for at least $45/share. So you enter this limit order: "SELL 100 ZING @ $45". A limit order to sell is called an "ask".

But you're not the only prospective seller of ZING stock. There's also an ask for $44/share and an ask for $46/share. If Jim Cramer wants to buy ZING, he's obviously going to pay as little as possible. So he'll take the $44 offer first, and only buy from you at $45 if he can't buy as much as he wants at $44.

The set of all standing bids and asks is called a "limit order book", or just a "book". You can buy a data feed from the stock market, and they will send you messages in real time telling you about changes to the book. Each message either adds an order to the book, or reduces the size of an order in the book (possibly removing the order entirely). You can record these messages in a log file, and later you can go back and analyze your log file.

Problem

Your task is to write a program, BookAnalyzer, that analyzes such a log file. BookAnalyzer takes one command line argument: *target-size*. BookAnalyzer then reads a market data log on standard input. As the book is modified, BookAnalyzer prints (on standard output) the total expense you would incur if you bought *target-size* shares (by taking as many asks as necessary, lowest first), and the total income you would receive if you sold *target-size* shares (by hitting as many bids as necessary, highest first). Each time the income or expense changes, it prints the changed value.

Input Format

The market data log contains one message per line (terminated by a single line feed character, '\n' in C or Java), and each message is a series of fields separated by spaces.

An "Add Order to Book" message looks like this:

timestamp A *order-id side price size*

Field	**Meaning**
timestamp	The time when this message was generated by the market, as milliseconds since midnight.
A	A literal string identifying this as an "Add Order to Book" message.
order-id	A unique string that subsequent "Reduce Order" messages will use to modify this order.
side	A 'B' if this is a buy order (a bid), and a 'S' if this is a sell order (an ask).
price	The limit price of this order.
size	The size in shares of this order, when it was initially sent to the market.

And a "Reduce Order" message looks like this:

timestamp R *order-id size*

Field	**Meaning**
timestamp	The time when this message was generated by the market, as milliseconds since midnight.
R	A literal string identifying this as an "Reduce Order" message.
order-id	The unique string that identifies the order to be reduced.
size	The amount by which to reduce the size of the order. This is not the new size of the order. If size is equal to or greater than the existing size of the order, the order is removed from the book.

The log file messages are sorted by timestamp by the time BookAnalyzer receives them.

Output Format

BookAnalyzer's output consists of one message per line, in the following format:

timestamp action total

Field	**Meaning**
timestamp	The timestamp from the input message that caused this output message to be generated.
action	A string: 'B' if this message contains the new expense to buy *target-size* shares, and 'S' if this message contains the new income for selling *target-size* shares.
total	The total expense (if *action* is 'B') to buy *target-size* shares, or the total income (if action is 'S') for selling *target-size* shares. If the book does not contain *target-size* shares in the appropriate type of order (asks for expense; bids for income), the total field contains the string 'NA'.

If BookAnalyzer encounters an error in an input message, it prints a warning to standard error and proceeds to the next message.

Example Input and Output

Here is an example run of BookAnalyzer with a *target-size* of 200:

Input	**Output**	**Notes**
28800538 A b S 44.26 100		No output yet because neither the bids nor the asks in the book have a total of 200 shares yet
28800562 A c B 44.10 100		Still not enough shares on either side of the book
28800744 R b 100		This reduces order 'b' to zero shares, which removes it from the book, so now the book contains no asks. But there's still no change to the total income or expense on 200 shares.

28800758 A d B 44.18 157	28800758 S 8832.56	The bid sizes now total 257, which is more than the target size of 200. To sell 200 shares, you would first hit the bid at 44.18 for 157 shares, spending $6936.26. Then you would hit the bid at 44.10 for the remaining 43 shares, spending another $1896.30. Your total income would be $8832.56
28800773 A e S 44.38 100		The book now contains a single ask of size 100, which is still not enough to change the target size expense from 'NA'
28800796 R d 157	28800796 S NA	This removes bid 'd' from the book, leaving just one bid with a size of 100 on the book, so the income from selling changes to 'NA
28800812 A f B 44.18 157	28800812 S 8832.56	This new bid brings the total bid size back over 200, so the selling income is no longer 'NA'
28800974 A g S 44.27 100	28800974 B 8865.00	This ask brings the total ask size up to 200, exactly the target size. The total expense for buying 200 shares would be 100 * $44.27 + 100 * $44.38 = $8865.00
28800975 R e 100	28800975 B NA	Removing ask 'e' from the book leaves less than 200 shares on the ask side, so the buying expense changes back to 'NA'
28812071 R f 100	28812071 S NA	Reducing bid 'f' by 100 shares leaves only 157 shares on the bid side, so the selling income changes to 'NA'
28813129 A h B 43.68 50	28813129 S 8806.50	This new bid makes it possible to sell 200 shares: 57 at $44.18, 100 at $44.10, and the last 43 at $43.68
28813300 R f 57	28813300 S NA	This removes bid 'f' from the book, so it is no longer possible to sell 200 shares
28813830 A i S 44.18 100	28813830 B 8845.00	This ask makes it possible to buy 200 shares again: 100 at $44.18 and 100 at $44.27

28814087 A j S 44.18 1000	28814087 B 8836.00	This ask has the same price as an existing ask, and these two asks are tied for the best asking price. This means you could now buy all 200 shares at $44.18 instead of buying half of them at $44.27, so the buying expense decreases
28814834 R c 100		This leaves only 50 shares on the bid side (all in order 'h'), so it is still not possible to sell 200 shares. The selling income is therefore unchanged from 'NA' and BookAnalyzer prints no output message
28814864 A k B 44.09 100		Only 150 shares on the bid side, so no output needed.
28815774 R k 100		Back to 50 shares on the bid side; still no output needed.
28815804 A l B 44.07 175	28815804 S 8804.25	There are now more than 200 shares on the bid side. You could sell 175 shares at $44.07 each, and the remaining 25 shares at $43.68 each
28815937 R j 1000	28815937 B 8845.00	After ask 'j' is removed from the book, you can still buy 200 shares: 100 at $44.18 each, and 100 at the worse price of $44.27
28816245 A m S 44.22 100	28816245 B 8840.00	Since $44.22 is a better price than $44.27, the buying expense decreases

Note that the book initially contains no orders, and that the buying expense and selling income are both considered to start at 'NA'. Since BookAnalyzer only produces output when the income or expense changes, it does not print anything until the total size of all bids or the total size of all asks meets or exceeds *target-size*.

What We're Looking For

Please write the BookAnalyzer program and send us the source code. You do not need to send us any compiler output or program output. You may use the language of your choice, unless directed otherwise. We encourage you to take advantage of the language's standard libraries, and in the case of C++ you may also use the Boost library. You cannot use other code that you would have to download separately from your language's normal distribution package.

We're looking for evidence that you can produce code that others would be able to understand, fix, and extend. At the same time, we do have real time requirements for production code, so we frown on gratuitous inefficiency. Write code the way you would on the job. Here are some qualities we look for:

- **Correctness** - Obviously, the fewer bugs you write, the fewer you'll have to fix.
- **Clarity** - If only you can understand your code, then only you can maintain it. Good code speaks for itself; a good programmer should be able to understand your implementation details without extensive comments.
- **Conciseness** - The less code you write, the less code your coworkers or your future self need to puzzle out.
- **Coefficiency** - (OK, just **Efficiency**, but wouldn't it be cool if all of these qualities started with a 'C'?) Anyway, using less time and space is generally better than otherwise.

Of course, there are often tradeoffs between each of these properties (except Correctness, which is pretty much an absolute). Also, we are a Linux shop and we like Unix style tools: programs that process the output of other programs. So make sure your implementation of BookAnalyzer is suitable for use in a shell pipeline. Follow the I/O specifications: don't mix prompts in with your output or demand that the input come from a disk file.

In addition to supplying us with your source code, please answer these questions:

- How did you choose your implementation language?
- How did you arrive at your final implementation? Were there other approaches that you considered or tried first?
- How does your implementation scale with respect to the target size?
- How does your implementation scale with respect to the number of orders in the book?

Test Data

You can download a larger amount of test input data in the file book_analyzer.in. This file is compressed using gzip. You should uncompress it before feeding it to your program. This is real market data, collected from a live system, so you might want to analyze it to decide what algorithms are most appropriate to use in BookAnalyzer. You can also download the corresponding output of our reference implementation of BookAnalyzer with various target sizes: 1, 200, 10000. These files are also compressed with gzip. In case you want to test your implementation on a smaller sample of data, here is a snippet in easy to cut and paste format:

```
28800538 A b S 44.26 100
28800562 A c B 44.10 100
28800744 R b 100
28800758 A d B 44.18 157
28800773 A e S 44.38 100
28800796 R d 157
28800812 A f B 44.18 157
28800974 A g S 44.27 100
28800975 R e 100
28812071 R f 100
```

28813129 A h B 43.68 50
28813300 R f 57
28813830 A i S 44.18 100
28814087 A j S 44.18 1000
28814834 R c 100
28814864 A k B 44.09 100
28815774 R k 100
28815804 A l B 44.07 175
28815937 R j 1000
28816245 A m S 44.22 100

And here is the corresponding output:

28800758 S 8832.56
28800796 S NA
28800812 S 8832.56
28800974 B 8865.00
28800975 B NA
28812071 S NA
28813129 S 8806.50
28813300 S NA
28813830 B 8845.00
28814087 B 8836.00
28815804 S 8804.25
28815937 B 8845.00
28816245 B 8840.00