Trading activities (the buying and selling of shares) take place on an exchange, such as the New York Stock Exchange (NYSE). One of the most important components of an exchange is known the **order book**. The order book is where investors' incoming "buy" and "sell" orders are centralized and matched with one another. When such a match between buyer and seller occurs, a trade takes place.

A decision to trade amounts to sending a buy or sell order to the market.
A **buy order** takes the following form: **Buy 10 shares of Apple at $150 or less**.
A **sell order** takes the following form: **Sell 10 shares of Apple at $150 or more**.

The "or less" component of the buy order simply means that this buyer is willing to pay a maximum of $150 for these 10 shares. Similar logic applies to the "or more" component of the sell order.
In finance terminology, the "or less" or "or more" for buy and sell orders, respectively, is always assumed. Therefore, "buy 10 shares Apple at 150" means "150 or cheaper" by default.

All orders are collected and displayed to the order book in sequence of price and time priority.

Price priority:
Buy orders with **higher quoted prices** are displayed **above** orders with lower prices. E.g. An order to buy 10 shares of Apple at $150 or less, as in the above example, will be given greater display priority than an order to buy 10 shares of Apple at $145 or less. The $150 order is more attractive to potential sellers, so this order is displayed above the $145 order.
Using the same logic, sell orders with **lower quoted prices** are displayed **above** orders with higher prices.

Time priority:
In the event that two orders arrive to the order book with the **same** quoted price (e.g. two buy orders for $150), the order that arrived first will be displayed above the one that arrived later.

When a new order arrives in the market, the exchange's matching algorithm will first see if it can **match** the order with an existing order in the order book (e.g. an incoming buy order for $150 will be matched immediately, and a trade takes place, if there currently exists a sell order for that same price or better). If there is no immediate match, the incoming order will be displayed in the order book according to the price and time priorities described above.

Finally, traders always have the option to **modify existing orders** or **delete orders**. Deleting obviously occurs when the trader is no longer interested in making the trade. Modify means: "The order I submitted a while ago to buy 10 Apple at 150, change it to buy 10 Apple at 155".

Assignment

Now let's look at an example.

In the morning, the order book is empty:

Buy			Sell		
Order ID	Size	Price	Price	Size	Order ID

Then, the first customer submits an order to buy 10 at the price of 100. The order book now looks like this:

Buy			Sell		
Order ID	Size	Price	Price	Size	Order ID
1	10	100			

Now, another customer places an order to sell 15 at the price of 105. This order cannot be matched with existing one, so it also goes to the order book:

Buy			Sell		
Order ID	Size	Price	Price	Size	Order ID
1	10	100	105	15	2

Now, one more customer places an order to buy 15 at the price of 100, and later someone submits an order to buy 10 at price of 95:

Buy			Sell		
Order ID	Size	Price	Price	Size	Order ID
1	10	100	105	15	2
3	15	100			
4	10	95			

And now, a customer sends an order to sell 20 at price of 100. Now, exchange can match this order with existing ones, causing 2 trades: order1 with order5 10@100 and order3 with order5 10@100. Mind that order 1 has priority over order 3 because it was submitted earlier. Here is how the book looks like now:

Buy			Sell		
Order ID	Size	Price	Price	Size	Order ID
3	5	100	105	15	2
4	10	95			

Now, let’s assume someone wants to sell 10@100. Exchange can match 5 shares or the incoming order with order 3, and the remainder (unmatched part of the new order) goes to the order book:

Buy			Sell		
Order ID	Size	Price	Price	Size	Order ID
4	10	95	100	5	6
			10	15	2

Now, your task is to implement such a matching algorithm.

Input: you submit orders from command line in the following format

a) Submit **new** order:
   N <B/S> size price
   For example
   N B 10 100

   Output: If there are immediate trades – output
   orderID of existing order | size | price
   If after the match there's an unexecuted portion that should go on top of book – output the new order's order ID for futher amendments and deletions.

b) **Modify** existing order
   M order_id size price
   Output: if there is a match – list trades as above
   Otherwise: just OK
   If there was an error: error message
   Please mind that if order is modified, it looses priority (i.e. it gets at the bottom of the queue for given price)

c) **Delete** existing order
   D order_id
   Output: OK or error

Good luck!