G: Parking Lot

Memory limit: **128 MB**

The marketplace of town M is a popular tourist attraction, but it's hard to get there without a car due to a very peculiar traffic policy. The traffic department decided to remove all trams and buses from town centre as they were impeding car traffic. But the cars require parking places which there aren't many of in the town centre. Thus, the small streets around marketplace were converted into very long parking lots for parallel parking. Young Johnny doesn't have a car yet but he's already vigorously advocating for this plan, because he got a job as a controller for one of the new parking lots. Unfortunately he's not so good at his new job, so he'll definitely need your help.

The street Johnny was appointed to, has length d. His job is to tell the approaching drivers where they should park their cars or inform them that their car is too long to fit. Formally he's supposed to serve two kinds of events:

- a car with length l_i and license plate p_i approaches,
- the car with license plate p_i leaves.

When a car approaches, Johnny has to find the shortest available contiguous unoccupied space where the car would fit. If there is more than one such place, he should choose the one that is closest to the marketplace (that is the earliest, looking from the start of parking lot). Car drivers are the masters of parallel parking and their vehicles are so swift that they always park at the very beginning of the space that Johnny finds; to the point that they can touch the preceding car, and in extreme situations they can even touch with both the preceding and the following car. Such tight parking is not a problem even when leaving the parking lot.

Input

The first line of input contains two integers d and q, separated by a single space, which denote the length of parking lot and the number of events respectively ($1 \le d \le 10^9$, $1 \le q \le 10^6$).

Each of the following q lines begins with a single letter `P` or `O`, which denote the kind of event (respectively approach and departure of a car), followed by a single space and a license plate p_i, which is a nonempty string of up to 10 characters, each of which is either small Latin letter (`a-z`) or a digit (`0-9`).

If the line describes approach of a car, then it also contains, after a single space, an (integer) length l_i ($1 \le l_i \le 10^9$) of the car.

Each car will try to find a parking place at most once: tourists never visit the marketplace more than once, this is even more true if they couldn't find parking place.

If the line describes departure of a car, then it contains nothing further. However, the license plate p_i is guaranteed to correspond to a car that already tried to find parking place. Note that the car didn't necessarily find the parking place and even in that case doesn't have to depart immediately (that is other cars can depart before it) or at all.

Output

The output should contain exactly q lines, each describing the result of the next event from input.

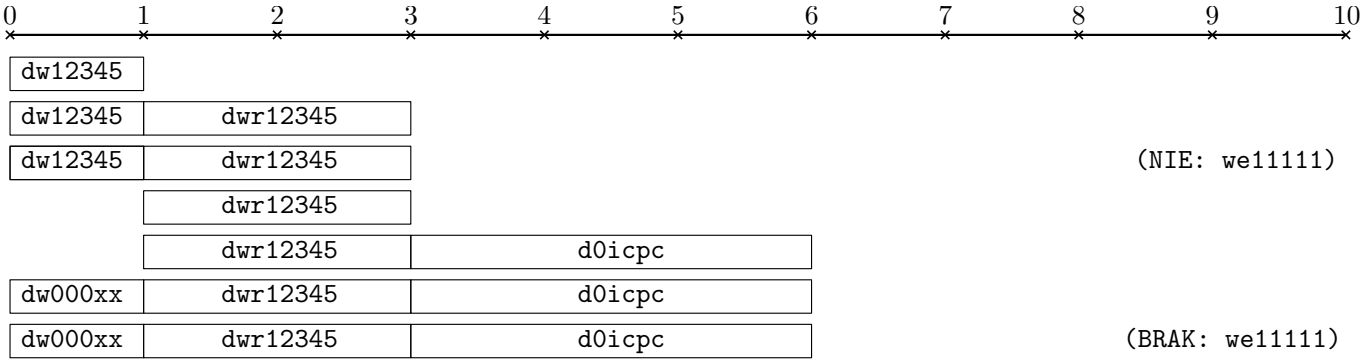
The result of an event describing the approach of a car is a single integer o_i – the position (counting from the beginning of parking lot) on which the car should be parked, or the word `NIE`, if it's not possible to park the car.

The result of an event describing the departure of a car is the word `OK`, if the car has actually parked and is now leaving or the word `BRAK` otherwise.

Example

Input	Output
10 7 P dw12345 1 P dwr12345 2 P we11111 8 O dw12345 P d0icpc 3 P dw000xx 1 O we11111	0 1 NIE OK 3 0 BRAK

The state of parking lot after each operation is shown in the following diagram.



The cars `dw12345` and `dwr12345` park one after another, at positions 0 and 1 respectively. The car `we11111` is too long and can't park. After the car `dw12345` departs, `d0icpc` parks just behind `dwr12345` at position 3, and then `dw000xx` parks at position 0, which has been made free by `dw12345`. Since `we11111` didn't find a parking place, the answer to its departure is `BRAK`.