Identification

You work for a government agency that is in charge of issuing important *identification numbers* for a new *social safety-net program (SSP)* that will benefit everybody in society (somehow, everyone is in full agreement on this). In order for the government to effectively run this program, you will need to issue a **SSP Number (SSPN)** to anybody who applies for one. There is a catch, however. It is better for individuals that are part of large organizations (like companies) to have *SSPNs* that are close to one another. Additionally, some people already have an *SSPN*, which is being carried over from another government program. Your task is to efficiently issue *SSPNs* to new applicants while carefully following the guidelines in this document.

First, *SSPNs* range from 0 to $2^n - 1$ (for some n). In addition, as stated earlier, some citizens already have an *SSPN* which you have been given ahead of time. These *SSPNs* cannot be changed for any individual and the *SSPNs* already issued have no set pattern or particular range of values (the last person in charge was issuing these pretty arbitrarily).

When a new individual applies for a new *SSP* number, you should follow these guidelines:

- If somebody applies as an *Individual*, issue them the smallest *SSPN* that is available to issue. This is the easiest case.
- If somebody applies as a member of a large organization, they can provide a *reference name* (of another member in that same organization). You should find the *SSPN* of the *reference name* and then issue the *SSPN* that is closest (in difference) to the given reference *SSPN*. Note that the newly issued *SSPN* could be smaller or larger than the *reference SSPN*. If the next available (smaller) *SSPN* is equidistant from the next available (larger) *SSPN* from the *reference SSPN*, then use the *available SSPN that is smaller than the reference SSPN.*
- If somebody is applying as the *first member of an organization*, you want to ensure that their *SSPN* is issued somewhere where there is a large block of available numbers (so that their future organization members are likely to get the numbers nearby). In this case, you should find the *largest contiguous block of unissued SSPNs* and issue the number that is in the middle of that block (rounded down). For example, if you determine that the largest contiguous block of unissued *SSPNs* is SSPN 5 through SSPN 25, you should issue *SSPN* $\lfloor \frac{(25+5)}{2} \rfloor = 15$. In general, use the formula $\lfloor \frac{(L+H)}{2} \rfloor$, where L is the lowest unissued number in the largest contiguous block, and H is the highest unissued number in the largest contiguous block. If there are multiple equally largest blocks of unissued numbers, than use the block containing the smallest *SSPNs*

It is also possible for an individual to apply to relinguish their *SSPN*. In this case, the person will provide you their name, and their *SSPN* will be relinquished for others to use.

Input

The first line of input contains the value $n \leq 20$. The *SSPNs* will all fall within the range $(0, 2^n - 1)$. The next line contains the number $I \leq 10^5$, the number of *SSPNs* that have already been issued / carried over from previous programs. The next I lines of input will contain the *names* and *SSPNs*, space separate and one per line, that have already been issued in no particular order. The next line will contain the number $Q \leq 10^6$, the number of applications for new / relinquished *SSPNs*. The next Q lines will contain the applications, one per line. Each of these lines will be in one of the following formats:

- The letter *I* (individual applicant) followed by the applicant's name (with no spaces).
- The letter *R* (applicant with reference) followed by the applicant's name (with no spaces) followed by name of the reference person who already has an issued *SSPN* in this applicant's organization. The applicant is requesting an *SSPN* close to *SSPN* of the given reference *name*.
- The letter *O* (organizational applicant) followed by the applicant's name (with no spaces).
- The letter *D* (delete) followed by the applicant's name.

Output

For each query that issues a new *SSPN* (i.e., the I, R, and O queries), output the applicant's full name (with no spaces, as it appeared in the input) and the new *SSPN* that will be issued to this applicant. If you run out of *SSPNs* to issue, simply print -1 next to the applicant's name instead. If an applicant is using a *reference SSPN* and the given *reference SSPN* has not been issued, then you should print -1 next to the applicant's name instead. You do not need to print anything out when the query involves relinquishing (deleting) a user's *SSPN*.

Notes

While we just studied Van-Emde-Boas trees, and that is the approach I hope you will at least consider taking, you are not required to use this particular data structure. Feel free to solve this problem however you wish! That being said, you are **required to build some kind of custom data structure**. You may not use a built-in data structure that already supports fast implementations of min, max, successor, predecessor, etc. We want you to gain practice playing around with various implementations of these ideas. The "model" solution we used for testing uses Van-Emde Boas Trees and was implemented in Python (so this is guaranteed to be feasible and pass in the required time limit). However, it is almost certainly the case that a wide array of other solutions are possible.

Writeup

For this assignment, you will also need to submit a **short writeup** to gradescope. This can be very informally. Please submit a pdf containing a paragraph or two summarizing how you solved the problem. We are specifically interested in the data structure component. How did you decide to implement the various operations you needed in order to solve the problem? What challenges arose? How did you address those challenges? We are mostly looking at this to ensure that you did indeed solve the problem with a custom data structure and that you did not use something built in to your language that does the heavy-lifting for you.

Sample Input

```
4
4
Mark 0
Maya 3
Sayu 4
Nick 8
4
I Floryan
R Varun Sayu
D Nick
O Vagul
```

Sample Output

```
Floryan 1
Varun 5
Vagul 10
```