CSC108H5F Fall 2022, Assignment 2

Due: To be submitted individually by Mon Oct 31, 22:00, on MarkUs

Please see the bottom of this handout for important assignment reminders.

Q1: Research Groups

In response to popular demand, Dan's established two research groups A and B composed of his `n` graduate students. Each of these students is a member of exactly one of the two groups at any given time. The students are numbered from `1` to `n` in some arbitrary order.

To measure the success of his research groups, Dan assigns a citation score to each of them. Initially, both groups have citation score 0. Throughout the semester, events of the following two types happen:

1. Student `x` publishes a paper in which they cite student `y`'s work (you can assume all of the students have been around long enough to have prior research that can be cited). As a result,
   - If `x` and `y` are in the same group, their group's citation score increases by 1.
   - If `x` and `y` are in different groups, `y`'s group's citation score increases by 5.
2. Dan changes student `x`'s group (from A to B or from B to A) to have them work on different projects.

To assess the overall success of the research groups, Dan needs to calculate their final citation scores at the end of the semester. Help him do so by writing a program that reports the final scores given a description of the events occurring throughout the semester.

Filename

Your filename for this question must be `q1.py`.

Input

The first line of the input contains two integers `n` and `m` separated by a single space. `n` denotes the number of Dan's students and `m` denotes the number of events.

The second line contains `n` letters separated by single spaces. The `i-th` letter is either "A" or "B". If it's "A", the `i-th` student is initially in group A; Otherwise, the `i-th` student is initially in group B.

The next `m` lines describe the events in chronological order. Each line describes a single event. If a line starts with the string `cite`, two integers `x` and `y` follow, indicating that student `x` has published a paper in which they cite student `y`'s work.
Otherwise, the line starts with the string `change`, which is followed by a single integer `x`. This indicates that Dan has changed student `x`'s group.

Output

Print two integers separated by a single space. These should indicate the final citation scores of groups A and B, respectively.

Sample Input 1

```
6 5
A A B B B B
cite 1 3
cite 1 2
change 1
cite 2 1
cite 4 2
```

Sample Output 1

```
6 10
```

Sample 1 Explanation

- The first line indicates that there are 6 students and 5 events.
- The second line indicates that the first two students are in group A while the remaining four are in group B.

- In the first event, student 1 cites student 3's work. Since they are in different groups, group B (student 3's group) scores 5 points.
- Student 1 cites student 2's work. Since they are both in group A, group A scores 1 point.
- Dan changes student 1's group (from group A to group B).
- Student 2 then cites student 1's work. Group B scores 5 points.
- Finally, student 4 cites student 2's work. Group A scores 5 points.
- In total, group A scores 6 points and group B scores 10 points, which is reflected in the output.

Sample Input 2

```
3 6
A A A
cite 3 1
change 1
cite 1 2
change 1
change 2
cite 2 1
```

Sample Output 2

```
11 0
```

Q2: Secret Courses

Dan's recently announced that he's teaching `n` top-secret courses next semester. Instead of enrolling in them through ACORN, students need to email Dan to express their interests. These courses are numbered from `1` to `n` in some arbitrary order.

In particular, if a student named `s` is interested in taking a course `c`, they need to send an email to Dan containing the message `c s`. Note that if a student is interested in taking multiple courses, they need to send multiple emails, one per course.

Upon receiving a message `c s`, Dan looks at the list of students already enrolled in course `c`. If there's already a student on the list whose name is *too similar* to `s`, Dan assumes `s` is the same student and ignores the message. Otherwise, he enrolls `s` in the course.

Dan considers two names *too similar* if and only if they have the same length and differ in at most one letter (note that "a" and "A" are considered the same letter). For example, "Josh" and "Josh" are *too similar*. "Sam" and "CaM" are *too similar* as well. However, neither "Max" and "Cat" nor "Ann" and "Anne" are *too similar*.

Dan has a lot of students and teaches a lot of courses. Consequently, it would take him forever to process the messages sent by the students one-by-one manually. Instead, he's asking you to help him out by writing a program that takes in the messages as the input and outputs, for every course, the list of the students enrolled in that course in the order of their enrolments.

Filename

Your filename for this question must be `q2.py`.

Input

The first line of the input consists of two space-separated integers `n` and `m`, denoting the number of secret courses Dan is teaching next semester and the number of messages sent by the students, respectively.

The `m` messages will be described in the following `m` lines in chronological order. The `i-th` line describes the `i-th` message and consists of an integer `c_i` followed by a string `s_i`. This indicates that a student named `s_i` wants to enrol in course `c_i`.

Output

You should output exactly `n` lines. The `i-th` line should contain the names of the students enrolled in the `i-th` course in the order of enrolment (note that the line would be empty if there were no students enrolled in the course). These names should be separated by single spaces.

Sample Input 1

```
2 6
1 alex
1 Alex
2 sam
1 alix
1 Alix
2 caM
```

Sample Output 1

```
alex
sam
```

Sample 1 Explanation

- The first line of the input indicates that there are 2 courses and 6 messages to process.
- Dan ignores the second, fourth, fifth, and sixth messages.
  - The second, fourth, and fifth messages are all ignored because “Alex”, “alix”, and “Alix” are all *too similar* to “alex”.
  - The sixth message is ignored because “caM” is *too similar* to “sam”.

Sample Input 2

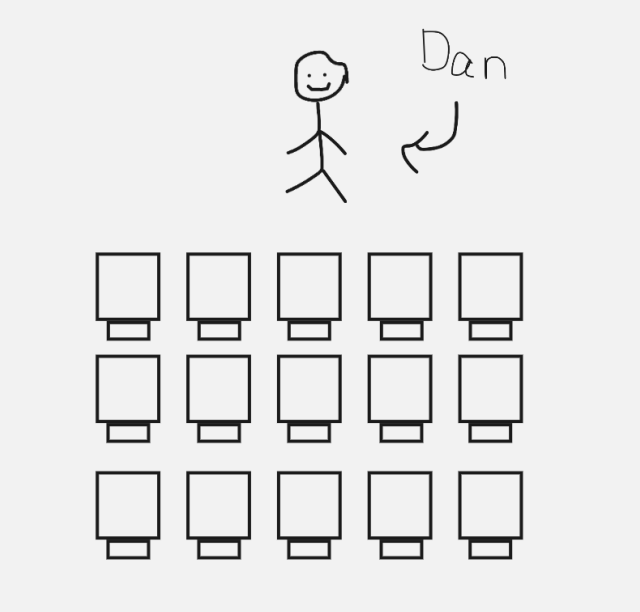
```
3 16
3 jun
3 Jin
1 Li
2 Kitty
2 Josh
3 Bob
1 Dave
2 Jose
1 David
3 Rob
3 Anne
3 Ann
2 Kevin
2 Lara
1 ALI
3 Xin
```

Sample Output 2

```
Li Dave David ALI
Kitty Josh Kevin Lara
jun Bob Anne Ann Xin
```

Q3: Lecture Hall

Dan holds his CSC108 lectures in a rectangular `N x M` lecture hall. In other words, this lecture hall has `N` rows of seats, each of them containing exactly `M` seats. Here's my attempt at drawing this layout when `N = 3` and `M = 5`:



lecture hall layout with 3 rows and 5 seats per row

The rows are numbered from `1` to `N` starting from the front row. Similarly, the columns are numbered from `1` to `M` starting from the leftmost column. We write `(r, c)` to denote the `c-th` seat in the `r-th` row.

When Dan walks into the lecture hall this morning, some of the seats are already taken (this is the initial layout of the lecture hall). After that, the students come in one group at a time. From experience, Dan knows that when a group of `K` students enter the lecture hall, they look for `K` consecutive empty seats. That is, they try to find an empty seat `(r, c)` such that for all integers `i` in `[0, K-1]`, the seat `(r, c + i)` exists and is empty. If they can't find `K` consecutive empty seats, they leave the lecture hall disappointedly and go play frisbee instead. Otherwise, they try to minimize `r` (if they have multiple options). If there are still multiple options with the minimum `r`, they try to minimize `c`. Note that with this description, one can always uniquely determine the seats they'll take given the layout of the seats and `K`.

Moreover, Dan observes that sometimes students that are already seated may get up and leave the lecture hall (maybe they were in the wrong class or Dan was putting them to sleep).

As you can imagine, it's distracting and noisy to have students walking around the lecture hall trying to find empty seats. Hence, as usual, Dan seeks to automate the process.

Write a program that given the initial layout of the lecture hall and a description of the above-mentioned events in the order they happen, reports where every group entering the lecture hall should take their seats.

Filename

Your filename for this question must be `q3.py`.

Input

The first line of the input contains two space-separated integers `N` and `M` denoting the number of rows and columns of the seat layout.

The initial layout of the seats is described in the next `N` lines. Each line contains a string of length `M` and describes a single row of seats. The front row is described by the first line, the second row is described by the second line, and so on. The `j-th` seat in the `i-th` row is initially occupied if the `j-th` character of the `i-th` line is `1` and not occupied if the character is `0`.

The next line contains a single integer `Q`, denoting the number of events. The events themselves are described in the following `Q` lines in chronological order. Each line describes a single event. If the event corresponds to `K` students entering the hall, the line consists of the string `in` followed by the integer `K`. Otherwise, the line consists of the string `out` followed by two integers `r` and `c`; this indicates that the student sitting at `(r, c)` leaves the lecture hall. It's guaranteed that there exists a student sitting there before this event.

Output

For every group of `K` students entering the lecture hall, report on a single line which seats they take. If they can't find `K` consecutive empty seats, print `-1`. Otherwise, if they take the seats `(r, c + i)` for `i` in `[0, K-1]`, print `r c`. Report these in the order they happen.

At the end of the output, print the final layout of the seats in the same format as the input.

Sample Input 1

```
3 4
0110
0001
1010
6
in 3
out 2 2
in 2
in 1
out 2 3
in 2
```

Sample Output 1

```
2 1
-1
1 1
2 2
1110
1111
1010
```

Sample 1 Explanation

- The first line of the input indicates that the lecture hall has 3 rows of 4 seats each.
- The initial layout of the lecture hall follows. Initially, five of the seats are taken.
- The next line indicates that 6 events occur during the lecture.
- A group of three students walk into the lecture hall. The only three consecutive empty seats are the first three seats in the second row. Hence, the three students take those seats. The layout is now

  ```
  0110
  1111
  1010
  ```

- The student sitting at `(2, 2)` leaves. The layout becomes

  ```
  0110
  1011
  1010
  ```

- A group of two students walk in. Since they can't find any two consecutive empty seats, they leave immediately.

- A single student walks in. They have multiple options for where to sit, so they try to sit as close to the front as possible. Row `1` has two empty seats, so the student will sit in row `1`. Of the two seats, the student chooses the leftmost one to minimize `c`. The layout becomes

  ```
  1110
  1011
  1010
  ```

- Two more events occur (make sure you understand their effect!).

Sample Input 2

```
2 4
1111
0000
11
in 3
out 1 1
in 2
out 2 3
in 2
out 2 1
out 2 2
out 1 2
in 2
in 2
in 10
```

Sample Output 2

```
2 1
-1
2 3
1 1
2 1
-1
1111
1111
```

Important Reminders

- You are not allowed to add any `import` statements.
- Test, test, test! The sample inputs/outputs that we give you in the assignment handout are not to be considered full testing.
- Please submit all of your files to MarkUs.
- Prior to submitting your files, it's in your best interest to try some other practice problems on DMOJ to get used to how to write and submit code. On DMOJ you paste your code, whereas on MarkUs you upload it, but the idea is otherwise similar.
- You must work alone on this assignment.
- The final thing you should do is to open your code files in Notepad and make sure that the code displays correctly as only Python code. This will help to ensure that you submit Python code files and not some other kind of file by mistake.