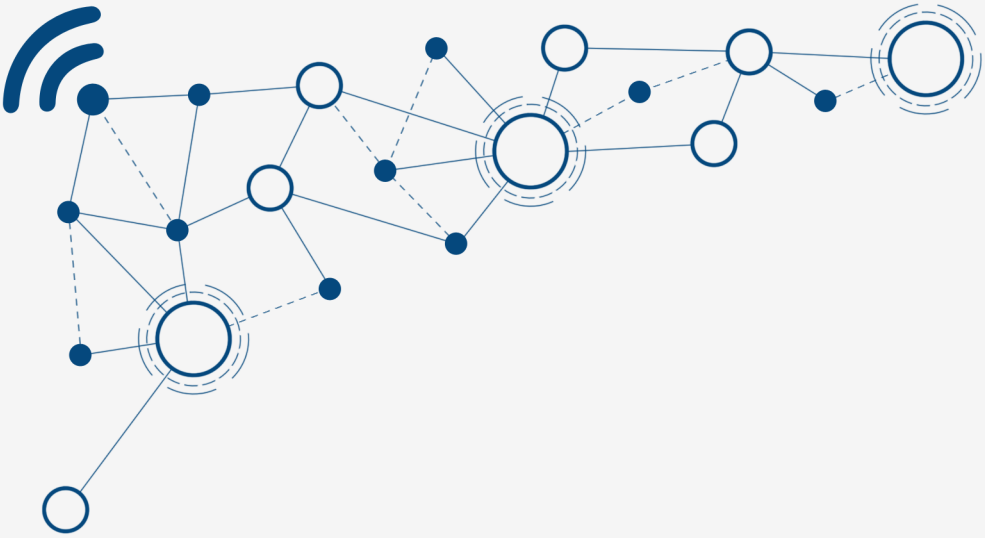
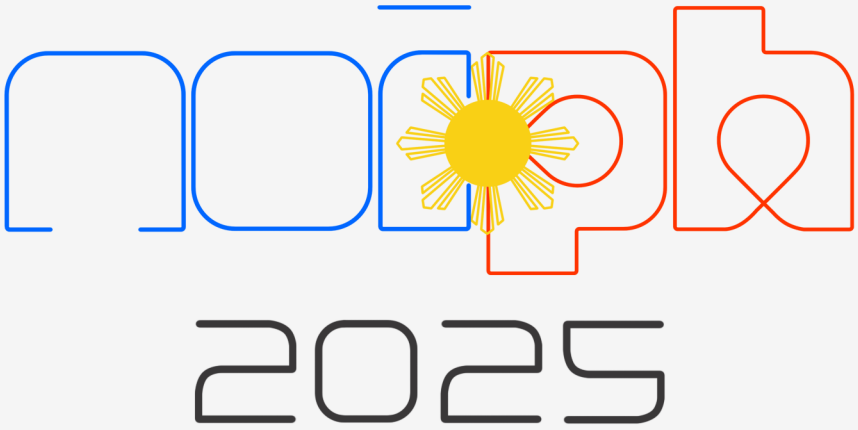
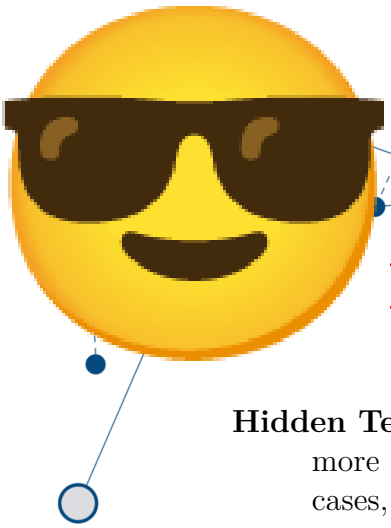




National Olympiad in Informatics

Finals Round 1



Important! Read the following:

Hidden Test Cases. Your solution will be checked by running it against one or more (usually several) hidden test cases. You will not have access to these cases, but a correct solution is expected to handle them correctly.

Strict Output Format. The output checker is **strict**. Follow these guidelines strictly:

- It is **space sensitive**. Do not output extra leading or trailing spaces. Do not output extra blank lines unless explicitly stated.
- It is **case sensitive**. So, for example, if the problem asks for the output in lowercase, follow it.
- Do not print any tabs. (No tabs will be required in the output.)
- Do not output anything else aside from what's asked for in the Output section. So, do not print things like "`Please enter t`".

Not following the output format strictly and exactly will likely result in the verdict "*Output isn't correct*".

Use Standard I/O. Do not read from, or write to, a file. You must read from the standard input and write to the standard output.

Submit Code Only. Only include **one** file when submitting: the source code (.cpp, .py, etc.) and nothing else.

No Java Package. For Java submissions, do not include a `package` line.

No Weird Filenames. Only use letters, digits and underscores in your filename. Do not use spaces or other special symbols.

Use Fast I/O. Many problems have large input file sizes, so use fast I/O. For example:

- In C/C++, use `scanf` and `printf`.
- In Python, use `sys.stdin.readline()`

Flush On Interactive Problems. On interactive problems, make sure to **flush** your output stream after printing.

- In C++, use `fflush(stdout);` or `cout << endl;`
- In Python, use `sys.stdout.flush()` or `print(flush=True)`
- For more details, including for other languages, ask a question/clarification through CMS.

Good luck and enjoy the contest! 😃

Contents

Notes

- Many problems have large input file sizes, so use fast I/O. For example:
  - In C/C++, use `scanf` and `printf`.
  - In Python, use `sys.stdin.readline()`
- On interactive problems, make sure to **flush** your output stream after printing.
  - In C++, use `fflush(stdout);` or `cout << endl;`
  - In Python, use `sys.stdout.flush()` or `print(flush=True)`
  - For more details, including for other languages, ask a question/clarification through CMS.

Good luck and enjoy the problems!

Problem B
Astig Runnings

The biathlon is a popular winter sport, combining cross-country skiing and rifle shooting. With the Philippines being a tropical country, if we wanted to join, we would definitely be at a disadvantage. However, inspired by the Jamaican Bobsleigh Team (and the Philippines' own history of sending alpine skiers to the Winter Olympics!), the PH Biathlon Team dares to dream.

The Philippine Biathlon Team has asked NOI.PH to help make an interesting training course for them, and so the following game was proposed.

There are m barriers lined up in a row.

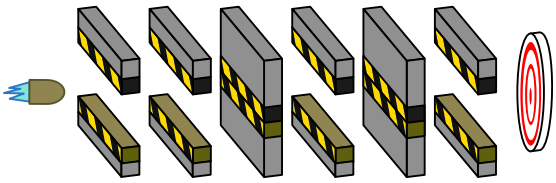
A biathlete is stationed at the far left. A target is stationed at the far right. All m barriers lie between the contestant and the target.

Each barrier can be either "closed" or "open". Contestants can see the starting configuration.

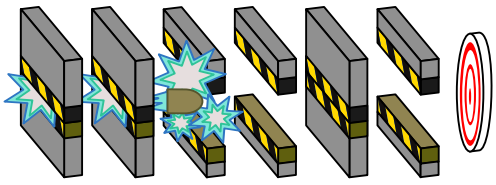
The contestant must fire their rifle in the direction of the target. But the barriers may be in the way!

- *If the bullet reaches an open barrier, it successfully passes through. The air pressure of the bullet then immediately causes the barrier to close behind it.*
- *If the bullet reaches a closed barrier, it strikes it! The bullet's path stops here, but the barrier becomes open as a result of the impact*

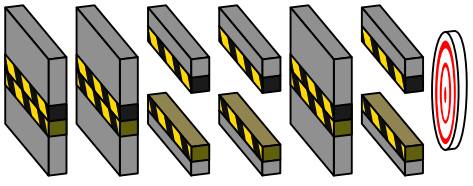
For example, consider the following configuration of barriers:



If they shoot the rifle once towards the target, here's what happens:



This would be the ultimate state of the barriers after that one shot:



The biathlete cannot hit the target until they manage to configure the barriers such that all of them are open before they take their shot (and the only way they can toggle this configuration... is also by just repeatedly firing their rifle). The contestant may fire their rifle multiple times, but they are limited by the number of bullets they have on them upon starting this course.

A contestant wins the game if they have enough bullets so that they can shoot the target.

This is a perfectly fine game, but it's just missing that NOI.PH *je ne sais quoi*. Of course we have to add a twisty twist. And what better way than to inject some randomness into the game?

We begin by constructing a *reference arena* with n barriers, labeled 1 through n from left to right, each either open or closed. When a biathlete arrives, we do the following:

- There are $n(n+1)/2$ pairs of indices (ℓ, r) such that $1 \leq \ell \leq r \leq n$; one of these is selected uniformly at random.
- Place them to the left of barrier ℓ, and the target to the right of barrier r.
- The biathlete plays the game from here, trying to shoot the target from their position (the barriers ℓ through r, inclusive, lie between the biathlete and the target; this means there are $m := r - \ell + 1$ barriers between them).
- Win or lose, after the game, barriers ℓ through r are **reset** to their original configuration.

The biathletes played our game q times. You are told how many bullets the playing biathlete had with them when they started each attempt. You are reminded that each attempt is independent from the others, since the barriers' configurations are restored when the game is over.

Help us determine, for each attempt, the probability that the biathlete wins. Specifically, for each attempt, just count the number of pairs of indices (ℓ, r) such that if this (ℓ, r) were chosen, then the biathlete would be able to shoot the target from those positions (given the number of bullets they have left).

Oh... So they're- This is just the skiing and shooting thing? I see... When you said you needed my help entertaining a group of rowdy biathletes, I thought you meant-

Input Format

The first line of input contains the two space-separated integers n and q.

The second line of input contains a string of length n, encoding the initial state of the barriers. The ith character is `O` if the ith barrier from the left is open, and `C` if it is closed.

This is followed by q lines, each describing an attempt by some biathlete. The t'th of these lines contains a positive integer b_t, the number of bullets the biathlete has in the t'th described attempt.

Output Format

Output q lines, the t'th of which should contain the answer for a biathlete who has b_t bullets.

Constraints

For all subtasks
$1 \leq n \leq 2 \times 10^5$ $1 \leq q \leq 3 \times 10^5$ $1 \leq b_t \leq 10^{18}$ for each t.

Subtask	Points	Constraints
1	**7**	$b_t = 1$ for all t
2	**6**	$b_t \leq 2$ for all t
3	**20**	All barriers in the reference arena are closed.
4	**18**	$n \leq 50$ $q \leq 50$
5	**24**	$n \leq 50000$ $q \leq 50$
6	**25**	No further constraints.

Sample I/O

Input 1	Output 1
7 3 OOCOCCC 3 2 21	12 9 23