Brainrot Blacklist

Some of the most annoying parts of working in a K–12 school are the jokes, slang, and meme culture that show up in what students look at online. The administration has been getting complaints from teachers who are tired of kids scrolling what they would call "brainrot" in class, and they have asked you to help with a *first* line of defense: flag video titles and search strings that **contain a word** on a school blocklist. The school's network is busy, so whatever you run on each request has to be **fast**, and the administration is **willing to live with an occasional wrong block** on a request that is actually clean, as long as the mechanism stays **light on memory** and the rate of that kind of mistake is **bounded** in a way they can tune.

A structure that stores every blocklisted string exactly (for example, a classic hash set keyed by the words themselves) would give perfect answers, but the deployment budget here points toward a **small-footprint, approximate** way to test whether a single *word* might be on the list: "definitely not" is reliable; "maybe yes" is allowed to happen sometimes for words that were never blocked, within a *design* limit that depends on a parameter in the input. **Part of the assignment** is to pick and implement a solution that matches that story—tight space, one-pass-style lookups per word, and a controllable false alarm rate for *word*-level tests—using what you have learned in this part of the course. You are not required to use any particular class name in your code, but you are expected to make a defensible, efficient choice.

What you are writing

You will read `n` **blocklist entries** and then `M_q` **request** lines. Each blocklist **entry** is a single string (one line of the file, read as a whole, even if it contains spaces; it is the thing you add once to your approximate set). Each request line can be a full title, empty, or anything else. Split each *request* line on ordinary ASCII whitespace (e.g. `split()` in Python with no extra arguments) into **word tokens** to test. **BLOCK** the request if *any* token is a possible blocklist match under your per-token approximate test. If *none* of the tokens are possible matches, the line is **ALLOWED**. If the request line has no tokens (empty, or only spaces and tabs), it is **ALLOWED**. A single title can therefore need many look-ups, and a false "maybe" on *one* word blocks the line—matching a real "any bad token anywhere in the string?" policy more closely than a single look-up on the full line.

Using `n` and ϵ from the input

The first line of input includes, among other things, the number of blocklisted words `n` and a value ϵ (a percent strictly between 5 and 30). You should use `n` and ϵ to **set up** your approximate per-word test so that, under the usual analysis of the method you choose, a word that was *never* inserted is reported as a possible match with probability about ϵ`/100` per look-up, while keeping the memory use proportional to a compact summary rather than a full exact table of size `n`. (How you do that is part of the technical design you are responsible for.)

Input

The first line has three integers: `n`, `M_q`, and ϵ.

- `n` is the number of blocklisted words, with $10 < n < 500$.
- `M_q` is the number of request lines, with $100 < M\_q < 10^5$.
- ϵ is a **target** false positive rate, in **percent**, for your **word-level** approximate tests, with $5 < \epsilon < 30$ (strictly between those numbers).

The next `n` lines each hold one **distinct** blocklist string (treat the whole line as a single key to insert, including any spaces). The next `M_q` lines are the requests to classify; a line may be empty.

Output

Print exactly `M_q` lines, one for each request in order. Each line is exactly either `ALLOWED` or `BLOCKED` as above.

Sample Input

```
10 15 5
skibidi
tung tung tung sahur
tralalero tralala
cappuccino ballerina
67
what the sigma
italian brainrot
gyatt
mewing
aura
POV: It's 3:00 AM and you hear tung tung tung sahur outside.
calculus 3 by 2blue2brown
He really thought he had gyatt...
I drank 67 cups of coffee
how to start mewing
full italian brainrot list 2025
completely normal text
an analysis on the word skibidi
mark floryan
You belong at UVA
keys on the kitchen counter
you might want a jacket.
How games fake water
tralalero tralala
what is what the sigma
```

Sample Output

```
ALLOWED
ALLOWED
ALLOWED
BLOCKED
BLOCKED
ALLOWED
ALLOWED
BLOCKED
ALLOWED
BLOCKED
ALLOWED
ALLOWED
ALLOWED
ALLOWED
BLOCKED
```

Checking your work

You can `diff` your program's output against the published `*.out` for each `*.in` in `io/` (same file stem). They were produced with one consistent reference; your job is to meet that behavior—approximate per-word test, sized from `n` and ϵ, and the token rules above—so your lines match.

A copy of the sample pairing lives in the repository as `io/sample_illustration.in` and `io/sample_illustration.out`.

Language

You may implement in C++, Java, or Python. Our written solution is in Python.