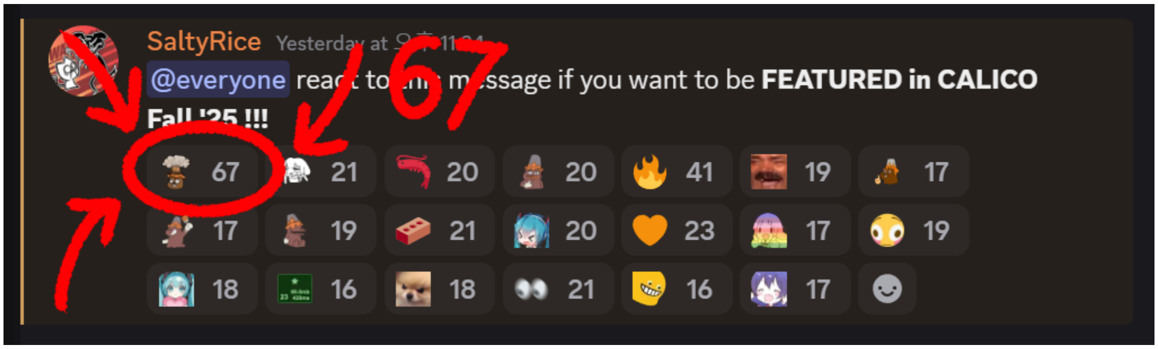
Problem 7: @everyone
8 Points

Problem ID: `reactions`
Rank: 2

Introduction

New CALICO announcement(s) just dropped! You want to react with a fire emoji (or a skull emoji), but you're taking the BART right now so your connection isn't the best. :/



SaltyRice Yesterday at 오후 11:

@everyone react to this message if you want to be **FEATURED in CALICO Fall '25 !!!**

Problem Statement

There are $\mathbf{A}$ Discord announcements in a Discord server that you'd like to try and react to. You can toggle a reaction to an announcement by sending a `REACT` for it, where the probability that Discord will receive any specific `REACT` is $\mathbf{P}_i$ for announcement i. In the case Discord does not receive a `REACT` that is sent, that `REACT` is ignored. Note that `REACT` *toggles* whether or not you've already reacted to an announcement—in other words, if Discord receives a `REACT` for an announcement that has already been reacted to, the reaction for that announcement will be removed. All $\mathbf{P}_i$ values are uniformly sampled from $[0.005, 1]$. Your goal is to react to at least 60% of the announcements.

Send any number of `REACT` for a single announcement before moving on to the next by sending `NEXT`. You will not know whether Discord receives any specific `REACT`. However, Discord always receives `NEXT`. For each announcement, you can send `REACT` and `NEXT` **combined** at most 500 times. Additionally, you can only send `REACT` and `NEXT` **combined** at most 10^6 times across all $\mathbf{A}$ announcements. Note that you have to send `NEXT` for all announcements, *including the last one*.

Input Format

The first line contains an integer $\mathbf{A}$ representing the total number of announcements. The next $\mathbf{A}$ lines each contain a single floating point number, the i^{th} of which represents $\mathbf{P}_i$.

Output Format

Starting with announcement $i = 1$, output any number of `REACT` before outputting `NEXT`. Each `REACT` or `NEXT` in the output should be on its own line.

Constraints

Time Limit: **2 Seconds**

$\mathbf{A} = 10^4$

$0.005 \leq \mathbf{P}_i \leq 1$ for all i

All $\mathbf{P}_i$ values are uniformly sampled from $[0.005, 1]$.

Sample Simulation

Note that this is not a sample test but rather a demonstration of the problem!

Sample Input

```
3
0.7
0.2
1.0
```

Sample Output

```
REACT
REACT
REACT
NEXT
REACT
REACT
NEXT
REACT
NEXT
```

Sample Example Output

Note that this is not a sample test! You will not be able to test your code on this sample!

A simulation for this sample input and output might look as follows:

In this example, we send `REACT` three times for the first announcement. For each `REACT`, the test judge samples a random number uniformly from $[0, 1]$. If that number is below $\mathbf{P}_0 = 0.7$, the `REACT` is received and the reaction on the announcement is toggled.

As a potential execution of this example:

1. On the first $\mathbf{REACT}$, we roll a 0.249, and successfully react to the announcement!
2. On the second $\mathbf{REACT}$, we roll a 0.748, the reaction fails, and nothing happens.
3. On the third $\mathbf{REACT}$, we roll a 0.509. The reaction succeeds, but since we already reacted to the announcement, we untoggle the reaction.
4. We then move onto the next announcement via $\mathbf{NEXT}$.

In this example, we send `REACT` two times for the second announcement. If the sampled random number is below $\mathbf{P}_1 = 0.2$, the `REACT` is received and the reaction on the announcement is toggled.

As a potential execution of this example:

1. On the first $\mathbf{REACT}$, we roll a 0.347, the reaction fails, and nothing happens.
2. On the second $\mathbf{REACT}$, we roll a 0.913, the reaction fails again, and nothing happens.
3. We then move onto the next announcement via `NEXT`.

In this example, we send one `REACT` for the third announcement. If the sampled random number is below $\mathbf{P}_2 = 1.0$, the signal is received and the emoji toggled.

As a potential execution of this example:

1. On the first $\mathbf{REACT}$, we roll a 0.593 and successfully react to the announcement!
2. We then move onto the next announcement via `NEXT`.

In this example execution, we have successfully reacted to 1 out of 3 announcements, yielding a final percentage for announcements reacted to of 33.333% (which is less than the target of 60%). As a result, this specific execution would cause the sample output shown to fail for the sample input.