Task 3: `Toxic Gene 2`

This is an interactive task. Note that only C++ is allowed for this task.

After a very successful career studying toxic bacteria, Benson the Rabbit has retired! James the Alien has taken over his lab and has recently discovered a new type of bacteria which he wants to investigate!

James has n species of bacteria, numbered 0 to $n-1$. Each of them falls into exactly one of 2 types: Regular or Toxic. t of them are Toxic. It is guaranteed that there is at least 1 Toxic and 1 Regular bacteria, i.e. $1 \leq t < n$. Note that James does not know the value of t.

James wants to identify the type of each bacteria. To analyze the bacteria, he bought a new machine to replace Benson's old one! This new machine allows James to place up to 1000 samples of bacteria in a row. Each sample consists of a single species, and multiple samples are allowed to be the same species.

After James has chosen the samples, the machine will run for a number of iterations. In each iteration, any Regular bacteria samples adjacent to a Toxic bacteria sample will die. The machine then shifts all surviving bacteria samples to fill the gaps such that the remaining bacteria form a single row.

The machine will run until the point where the next iteration would have the same number of surviving bacteria as the current iteration. This means that the machine will always run at least 1 iteration, even if no bacteria die in the first iteration.

The machine will then report to James the number of surviving bacteria samples after every iteration. This forms a single query. Each use of the machine takes time, and James does not have a lot of time. He may only use the machine up to 1000 times. Help James determine for each bacteria species, whether it is Regular or Toxic in as few queries as possible.

Implementation Details

This is an interactive task. Do not read from standard input or write to standard output.

You are to implement the following function.

- `void determine_type(int n)`

This function will be called exactly once per testcase. The function will be passed, n, representing the number of bacteria species James has.

You are allowed to call the following grader functions to complete the task:

- `std::vector<int> query_machine(std::vector<int> samples)`
- `void answer_type(std::vector<char> v)`

The function `query_machine` is called with a 1-dimensional array `samples`, describing the species of bacteria you place in the machine, in the order that you place them. The size of `samples` cannot be more than 1000, and must contain integers from 0 to $n - 1$. Additionally, the vector passed to `query_machine` **will not be modified**. In other words, you can expect the vector `samples` to remain the same after calling `query_machine`.

The function will return a vector representing the number of surviving bacteria samples after each iteration until the machine stops running.

The function `answer_type` must be called with a vector of exactly n characters, with the i-th character representing the type of i-th species of bacteria. Each character in the vector may be equal to 'R' or 'T', representing Regular and Toxic bacteria respectively.

The following situations will cause you to receive a *Wrong Answer* verdict and cause the program to terminate immediately:

- If `query_machine` is called more than 1000 times or with more than 1000 samples
- If `answer_type` is called more than once
- If the vector passed into `answer_type` does not have exactly n elements, or has incorrect bacteria types
- If `query_machine` is called after a call to `answer_type`
- If `answer_type` has not been called by the time `determine_type` terminates

Do note that the grader for this task is **not adaptive**. This means that the answer for each testcase is fixed and will not change during the execution of your program.

Sample Interaction

In this example only, $n = 3$. In all other test cases $n = 1000$. Your program does not need to solve this sample to be considered correct.

Suppose that, bacteria species 0 is Toxic, while bacteria species 1 and 2 are Regular. Your function will be called with the following parameters:

```
determine_type(3)
```

A possible interaction could be as follows:

- `query_machine([1, 0, 2, 1]) = [2, 1]`

  This call represents putting 1 sample each of species 0 and 2, and 2 samples of species 1, in the order `[1, 0, 2, 1]`.

  During the first iteration, the first sample, of species 1, and the third sample, of species 2, are killed since they are Regular and are adjacent to a Toxic species, 0. After this, there are 2 surviving bacteria samples, `[0, 1]`.

  During the second iteration, the second sample, of species 1, is killed. After this, there is 1 surviving bacteria sample, `[0]`. At this point, any future iterations will also have 1 surviving bacteria sample, so the machine terminates and returns the number of surviving samples after each iteration: `[2, 1]`.

- `query_machine([1, 0, 1, 0, 0]) = [3]`

  During the first iteration, the first and third sample, of species 1, will both die. After this, there are 3 surviving bacteria samples, `[0, 0, 0]`. At this point, any future iterations will also have 3 surviving bacteria sample, so the machine terminates and returns: `[3]`.

- `query_machine([2, 1]) = [2]`

  During the first iteration, none of the bacteria die. At this point, any future iterations will also have 2 surviving bacteria sample, so the machine terminates and returns: `[2]`

At this point, the program decides that it has enough information to determine the types of bacteria and makes the following call:

- `answer_type(['T', 'R', 'R'])`

This call does not return any value. As the program has correctly identified all $n = 3$ bacteria species types, used no more than 1000 queries, and not made any invalid calls, it will be considered correct for this test case.

Subtasks

Your program will be tested on input instances that satisfy the following restrictions:

- $n = 1000$
- $1 \leq t < 1000$

Subtask	Marks	Additional Constraints
0	0	Sample Testcases
1	10	Bacteria 0 is toxic
2	90	No additional constraints

Scoring

Subtasks 2 is a partial scoring subtask. Your score depends on the number of queries you make across all testcases in that subtask, which we will denote here as q.

- If $q > 1000$, your score is 0.
- If $21 < q \leq 1000$, your score is $90 \times \sqrt{\frac{21}{q}}$
- If $q \leq 21$, your score is 90.

Testing

You may download the grader file, the header file and a solution template under *Attachments*. Two input files are provided for your testing, `sample1.txt` corresponds to the testcase in the sample interaction, and `sample2.txt` contains a testcase with $n = 1000$. Please use the script `compile.sh` to compile and run your solution for testing.

Grader Input Format for Testing

The first line contains one integer n, the number of bacteria species.

The next line contains a string of n charaters ('R' or 'T'), describing the types of all species of bacteria in order.

Grader Output Format for Testing

The result of the call to `determine_type` is printed on a single line.

If your program enters any of the situations that triggers a *Wrong Answer* verdict, the grader prints the corresponding *Wrong Answer* message and terminates immediately.

Otherwise, the grader outputs the number of calls made to `query_machine`. Note that unlike the actual grader, the sample grader does not terminate immediately if more than 1000 calls to `query_machine` are made.