Task 5: `Toxic Gene`

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

Benson the Rabbit's plane has been overwhelmed by toxic bacteria, and he has to investigate it!

Benson the Rabbit has n species of bacteria. Each of them falls into exactly one of 3 types: Regular, Strong, Toxic. t of them are Toxic. It is guaranteed that there is at least 1 Toxic bacteria, i.e. $t \geq 1$. Note that Benson does not know the value of t.

Benson wants to identify the type of each bacteria. To analyze the bacteria, he can place bacteria specimens into a machine. He can specify the species of each bacteria, and he can add any number of each species into the machine, including 0. This forms a single sample. Due to size constraints, the total number of bacteria in a sample cannot exceed 300.

Each of the 3 types of bacteria have the following properties:

- Regular bacteria will survive if there are no Toxic bacteria in the sample, and will die if there is at least one Toxic bacteria in the sample.
- Strong bacteria will always survive.
- Toxic bacteria produce a toxin which kills all bacteria in the sample that are not Strong bacteria. Toxic bacteria will always die.

After a sample is selected, the machine will tell Benson how many bacteria survived in total. Each use of the machine takes time, and Benson does not have a lot of time. He may only use the machine up to 600 times. Help Benson determine for each bacteria, whether it is Regular, Strong or Toxic in as few samples 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 up to 100 times per testcase. Each call may have a different combination of Regular, Strong and Toxic bacteria. For each testcase, you must solve all calls to the function within the time and memory limits.

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

- `int query_sample(std::vector<int> species)`
- `void answer_type(int x, char c)`

The function `query_sample` is called with a 1-dimensional array `species`, describing the species of each of the bacteria in the sample your program has chosen. The size of `species` cannot be more than 300. Additionally, the array passed to `query_sample` **will not be modified**. In other words, you can expect the array `species` to remain the same after calling `query_sample`.

The function `answer_type` is called with one integer `x` and one character `c`. Your program may call this function when it is sure that the species `x` is of the type represented by `c`. `c` may be equal to `'R'`, `'S'` or `'T'`, representing Regular, Strong and Toxic bacteria types respectively. Your program must call this function for all species of bacteria.

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

- If either `query_sample` or `answer_type` is called with invalid parameters
- If `answer_type` identifies the type of bacteria wrongly
- If by the time `determine_type` terminates, not all n species of bacteria have been identified via the `answer_type` function.
- If `query_sample` is called more than 600 times during a call of `determine_type`

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

Suppose $n = 5$. Bacteria species 1 and 2 are Toxic, bacteria species 3 and 4 are Regular and bacteria species 5 is Strong. This corresponds to the string `TTRRS`.

Your function will be called with the following parameters

```
determine_type(5)
```

A possible interaction could be as follows:

- `query_sample([1,2,3,4,5]) = 1`

  One specimen of each bacteria is used in this query. Only bacteria 5 survives, so the grader returns the value 1.

- `query_sample([3,3,4,5]) = 4`

  Two specimen of bacteria species 3 and one specimen each of bacteria species 4 and 5 are used in this query. Since there are no toxic bacteria, all specimens survive and the grader returns the value 4.

At this point, the program decides that it has enough information to determine the type of each bacteria and makes the following 5 calls.

- `answer_type(1,'T')`
- `answer_type(2,'T')`
- `answer_type(3,'R')`
- `answer_type(4,'R')`
- `answer_type(5,'S')`

None of these calls return any value. As the program has correctly identified all $n = 5$ bacteria species types and used no more than 600 queries, it will be considered correct for this test case.

Do note that this sample test case does not satisfy the input limits below. It is purely for testing purposes and is not required to be solved to obtain full points for this task.

Scoring

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

- $n = 300$
- $1 \leq t \leq 30$

Your score for this task depends on the maximum number of queries you make across all calls to `determine_type` over all testcases, which we will denote here as m.

- If $m > 600$, your score is 0.

- If $340 < m \leq 600$, your score is $2 + 7 \times \frac{600-m}{260}$.
- If $275 < m \leq 340$, your score is $9 + 15 \times \frac{340-m}{65}$
- If $190 < m \leq 275$, your score is $24 + 22 \times \frac{275-m}{85}$
- If $150 < m \leq 190$, your score is $46 + 54 \times \frac{190-m}{40}$.
- If $m \leq 150$, your score is 100.

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 $tc = 100$ and $n = 300$. Please use the script `compile.sh` to compile and run your solution for testing.

Grader Input Format for Testing

The first line contains two integers tc and n. tc is the number of calls to `determine_type`.

Each of the next tc lines contain a string of n characters (`R`, `S` or `T`) describing the types of all species of bacteria in order.

Grader Output Format for Testing

The result of each call to `determine_type` is represented as an integer, printed on a new line.

If your program enters any of the situations that triggers a *Wrong Answer* verdict, the grader outputs `-1` and terminates immediately. Any remaining calls to `determine_type` will not be made.

Otherwise, the grader outputs the number of calls made to `query_sample`.