Ancient Machine

Anna and Bruno are archaeologists. They excavate ruins of IOI Kingdom. In the ruin A, Anna discovered the layout of an old machine. In the ruin B, Bruno discovered the actual machine.

This machine consists of N devices. The devices are attached to an electric wire in a row. There are three types of devices called `X`, `Y`, `Z`. From the left, the devices are numbered from 0 through $N-1$. The type of the device i ($0 \leq i \leq N-1$) is S_i. In other words, S_i is either `X`, `Y`, or `Z`.

Since the machine is too big, Bruno decided to remove the devices, one by one, from the machine. However, since the devices interact with each other by an electric wire, he must be very careful about the order of removal.

Concerning a way to remove a device from the machine, we define the following.

- Assume that the devices x, y, z ($0 \leq x < y < z \leq N-1$) are not yet removed, and $S_x =$ `X`, $S_y =$ `Y`, $S_z =$ `Z`. Moreover, assume that, for every $x < j < y$, the device j was already removed, and, for every $y < k < z$, the device k was already removed. If all of these conditions are satisfied and we remove the device y from the machine, it is called a **good removal**.
- Any other way to remove a device from the machine is not a good removal.

Bruno has to remove all of the N devices from the machine so that the number of good removals is the maximum possible. However, since the three types of devices look similar, he cannot distinguish the types of the devices.

Since Anna has the layout of the machine, she knows the type of each device attached to the machine. Thus, she will use a transmitter to help Bruno. Using the transmitter, she can send a sequence of characters. Each character she can send is either 0 or 1.

Write a program which implements Anna's strategy and Bruno's strategy so that the number of good removals is the maximum possible. In this task, if the number of characters sent by Anna to Bruno is smaller, your will get higher score.

Implementation Details

You need to submit two files.

The first file is `Anna.cpp`. It should implement Anna's strategy. It should implement the following function. The program should include `Anna.h` using the preprocessing directive `#include`.

- `void Anna(int N, std::vector<char> S)`
  For each test case, this function is called exactly once in the beginning.

- The parameter `N` is the number of devices N.
- The parameter `S` is an array of length N. This means `S[i]` is the type S_i of the device i ($0 \leq i \leq N-1$). Here the character `S[i]` is either '`X`', '`Y`', or '`Z`'.

Your program can call the following function.

★ `void Send(int a)`

Using this function, Anna sends a character, 0 or 1, to Bruno.

- The parameter `a` is an information sent to Bruno. It should be 0 or 1. If this condition is not satisfied, your program is judged as **Wrong Answer [1]**.
- The function `Send` should not be called more than 200 000 times. If it is called more than 200 000 times, your program is judged as **Wrong Answer [2]**.

The second file is `Bruno.cpp`. It should implement Bruno's strategy. It should implement the following function. The program should include `Bruno.h` using the preprocessing directive `#include`.

- `void Bruno(int N, int L, std::vector<int> A)`

After the function `Anna` is called, this function is called exactly once.

- The parameter `N` is the number of devices N.
- The parameter `L` is the number L of characters, 0 or 1, sent by Anna.
- The parameter `A` is an array of length L. It means Anna sent the sequence `A[0],A[1],...,A[L-1]` of characters to Bruno, in this order. Each character of the sequence is either 0 or 1.

Your program can call the following function.

★ `void Remove(int d)`

Using this function, your program answers how to remove the devices.

- The parameter `d` is the index of a device. It means Bruno removes the device d.
- The inequalities $0 \leq d \leq N-1$ should be satisfied. If this condition is not satisfied, your program is judged as **Wrong Answer [3]**.
- It is not allowed to call this function with same parameter d more than once. If this condition is not satisfied, your program is judged as **Wrong Answer [4]**.
- The function `Remove` should be called exactly N times. When the function `Bruno` terminates, if the number of calls to the function `Remove` is different from N, your program is judged as **Wrong Answer [5]**.
- After all the N devices are removed, the number of good removals should be the maximum possible. If this condition is not satisfied, your program is judged as **Wrong Answer [6]**.

Important Notices

- Your program can implement other functions for internal use, or use global variables. Submitted files will be compiled with the grader, and become a single executable file. All global variables and internal functions should be declared in an unnamed namespace to avoid confliction with other files. When it is graded, it will be executed as two processes of Anna and Bruno. The process of Anna and the process of Bruno cannot share global variables.
- Your program must not use the standard input and the standard output. Your program must not communicate with other files by any methods. However, your program may output debugging information to the standard error.

Compilation and Test Run

You can download an archive file from the contest webpage which contains the sample grader to test your program. The archive file also contains a sample source file of your program.

The sample grader is the file `grader.cpp`. In order to test your program, put `grader.cpp`, `Anna.cpp`, `Bruno.cpp`, `Anna.h`, `Bruno.h` in the same directory, and run the following command to compile your programs.

```
g++ -std=gnu++17 -O2 -fsigned-char -o grader grader.cpp Anna.cpp Bruno.cpp
```

When the compilation succeeds, the executable file `grader` is generated.

Note that the actual grader is different from the sample grader. The sample grader will be executed as a single process, which will read input data from the standard input and write the results to the standard output.

Input for the Sample Grader

The sample grader reads the following data from the standard input.

N
$S_0\ S_1\ \cdots\ S_{N-1}$

Here S_i and S_{i+1} $(0 \leq i \leq N-2)$ are separated by a space.

Output of the Sample Grader

When the program terminates successfully, the sample grader writes the following information to the standard output (quotes for clarity).

- If your program is judged as any one of Wrong Answer [1], [2], [3], [4], or [5], it writes its type as "`Wrong Answer [1]`".
- Otherwise, it writes the number L of function calls to `Send` and the number D of good removals as "`Accepted: L D`". Note that the behavior of the sample grader is different from the actual grader. The sample grader does not check whether your program is judged as Wrong Answer [6] or not.

If your program is judged as several types of Wrong Answer [1], [2], [3], [4], or [5], the sample grader reports only one of them.

Constraints

- $3 \leq N \leq 100\,000$.
- S_i is one of `X`, `Y`, or `Z` ($0 \leq i \leq N-1$).

Subtasks

1. (5 points) $N \leq 18$.
2. (95 points) No additional constraints. In this Subtask, your score is calculated by the following way.
   - Let L be the maximum of the number of function calls to `Send` for every test case of this Subtask.
   - Then your score is calculated as follows.
     - If $160\,000 < L \leq 200\,000$, your score is $25 + \left\lfloor 10 \times \frac{200000 - L}{40000} \right\rfloor$ points.
     - If $100\,000 < L \leq 160\,000$, your score is $35 + \left\lfloor 30 \times \frac{160000 - L}{60000} \right\rfloor$ points.
     - If $70\,000 < L \leq 100\,000$, your score is $65 + \left\lfloor 30 \times \left(\frac{100000 - L}{30000} \right)^2 \right\rfloor$ points.
     - If $L \leq 70\,000$, your score is 95 points.

   Here $\lfloor x \rfloor$ is the largest integer not exceeding x.

Sample Communication

Here is a sample input for the sample grader and corresponding function calls.

Sample Input 1	Sample Function Calls	
	Call	Call
4 X Y X Z	`Anna(4, {X, Y, X, Z})`	
		`Send(0)`
		`Send(1)`
	`Bruno(4, 2, {0, 1})`	
		`Remove(2)`
		`Remove(1)`
		`Remove(0)`
		`Remove(3)`

In these sample function calls, the 4 devices will be removed by the following way.

1. In the beginning, the 4 devices are `X Y X Z`.
2. Bruno removes the device 2. The devices become `X Y - Z`. Here - means the device in this position was already removed.
3. Bruno removes the device 1. The devices become `X - - Z`. Since $(x, y, z) = (0, 1, 3)$ satisfies the condition, it is a good removal.
4. Bruno removes the device 0. The devices become `- - - Z`.
5. Finally, Bruno removes the device 3. The devices become `- - - -`.

The number of good removals is 1.
In this sample input, the number of good removals cannot be greater than 1.