Fortune Telling 3

Anna and Bruno like fortune-telling and enjoy playing various styles of fortune-telling together. Today, they will play fortune-telling using cards, which is described below. They will play it for Q times.

1. They prepare many cards with 0 or 1 written on each one, shuffle them, and pile them up on the deck.
2. Anna draws N (= 900) cards from the deck, one at a time. Anna and Bruno know the value of N. Every time she draws a card, she decides whether to discard the card or put the card on the table.
   - If she selects to put the card on the table, she inserts the card into the sequence of cards on the table.
   - More formally, when there are l cards on the table, she designates a non-negative integer x ($0 \leq x \leq l$) and puts the card immediately right to the x-th leftmost card on the table. In the case of $x = 0$, she puts the card on the leftmost of the sequence of cards on the table.
3. The procedure for Anna finishes when she draws and processes N cards. The result of the fortune-telling is the number of cards with 1 among the N cards.
4. After Anna finishes her procedure, Bruno sees the sequence of cards on the table. Using the information, he needs to guess the result of the fortune-telling. If he guesses correctly, the fortune-telling is a success.

The fortune-telling is considered more proficient when fewer cards are on the table. Write a program to implement Anna's and Bruno's strategies to succeed in all Q fortune tellings. In this task, the fewer cards Anna puts on the table, the higher the score will be.

Implementation Details

You should submit two files.

The first file is `Anna.cpp`. This file is intended to implement Anna's strategy and should implement the following function. The program should include `Anna.h` using the preprocessor directive `#include`.

- `void Anna(int N)`

  This function is called Q times. The i-th call ($1 \leq i \leq Q$) of this function corresponds to Anna's procedure in the i-th fortune-telling.

  - The parameter `N` is the number of cards that Anna draws.

For each call of the function `Anna`, this function must call the following function $N + 1$ times.

★ `int DrawCard(int x)`

Using this function, Anna draws the card from the deck, and selects whether to discard it or put it on the table. She obtains the number written on the j-th card via the return value of the j-th call ($1 \leq j \leq N$) of this function. She designates the operation for the $(k-1)$-th card via the parameter for the k-th call ($2 \leq k \leq N+1$) of this function.

- The return value of the j-th call ($1 \leq j \leq N$) of this function is 0 or 1, representing the number written on the j-th card.
- The return value of the $(N+1)$-th call of this function is -1, representing that Anna finished drawing cards.
- The parameter x for the 1-st call of this function must satisfy $x = -1$. If not, your program will be judged as **Wrong Answer [1]**.
- The parameter x for the k-th call of this function represents the operation for the $(k-1)$-th card. If $x = -1$, the $(k-1)$-th card will be discarded. If $0 \leq x$, the $(k-1)$-th card will be put immediately right to the x-th leftmost card on the table. In the case of $x = 0$, the card will be put on the leftmost of the sequence of cards on the table. Here, the parameter must satisfy $-1 \leq x \leq l$, where l is the current number of cards on the table. If not, your program will be judged as **Wrong Answer [2]**.
- If the number of calls of the function DrawCard is not $N+1$, your program will be judged as **Wrong Answer [3]**.

The second file is `Bruno.cpp`. This file is intended to implement Bruno's strategy and should implement the following function. The program should include Bruno.h using the preprocessor directive `#include`.

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

This function is called exactly once after each call of the function Anna, totaling Q times. The i-th call ($1 \leq i \leq Q$) of this function corresponds to Bruno's procedure in the i-th fortune-telling. It must return the number of cards with 1 among the cards Anna drew.

- ◇ The parameter `N` is the number of cards that Anna drew.
- ◇ The parameter `L` is the number of cards on the table.
- ◇ The parameter `C` is the array of length L, where `C[l-1]` is the number written on the l-th leftmost card ($1 \leq l \leq L$) on the table.
- ◇ If the return value of the function Bruno is not equal to the number of cards with 1 among the cards Anna drew, your program will be judged as **Wrong Answer [4]**.

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 method. 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++20 -O2 -o grader grader.cpp Anna.cpp Bruno.cpp
```

Instead, you may run `compile.sh` contained in the archive file.

```
./compile.sh
```

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 and the standard error output.

Input for the Sample Grader

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

$$
\begin{aligned}
&Q\ N\\
&A_{1,1}\ A_{1,2}\ \cdots\ A_{1,N}\\
&A_{2,1}\ A_{2,2}\ \cdots\ A_{2,N}\\
&\vdots\\
&A_{Q,1}\ A_{Q,2}\ \cdots\ A_{Q,N}
\end{aligned}
$$

$A_{i,j}$ $(1 \leq i \leq Q, 1 \leq j \leq N)$ is the number written on the j-th card Anna draws in the i-th fortune telling.

Output of the Sample Grader

The sample grader outputs the following information to the standard output and the standard error output (quotes for clarity).

- If the program is judged as correct, the sample grader outputs the maximum value of the number of cards on the table in the format "`Accepted: 100`".
- If the program is judged as wrong answer, the sample grader outputs the type of wrong answer in the format "`Wrong Answer [1]`".

If your program meets the conditions of several types of wrong answer, the sample grader reports only one of them. The sample grader may terminate the execution when one of the conditions for wrong answer is met.

Grading Considerations

The actual grader is **not** adaptive for all testcases. It means that the sequence of numbers written on the cards is fixed before the program's execution.

Constraints

All the input data satisfy the following conditions.

- $1 \leq Q \leq 100$.
- $N = 900$.
- $A_{i,j}$ is either 0 or 1 ($1 \leq i \leq Q$, $1 \leq j \leq N$).

Grading

If your program is judged as any type of **Wrong Answer [1] – [4]** (see Implementation Details), Time Limit Exceeded, Memory Limit Exceeded, or Runtime Error, in any testcase, your score is 0 points.

If your program is judged as correct for all testcases, your score is determined by the maximum number of calls for the function `DrawCard` with parameter $0 \leq x$, among all plays of the fortune-telling for all testcases. Let L be this number. Your score will be:

- $500 < L$: 3 points
- $14 < L \leq 500$: $\left\lfloor 100 \times \left(\frac{2.5}{L - 11.5} \right)^{0.35} \right\rfloor$ points
- $L \leq 14$: 100 points

Sample Communication

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

Sample Input 1	Sample Function Calls			
	Call	Return Value	Call	Return Value
2 5 0 1 0 0 1 1 1 0 1 0	Anna(5)			
			DrawCard(-1)	0
			DrawCard(0)	1
			DrawCard(-1)	0
			DrawCard(1)	0
			DrawCard(-1)	1
			DrawCard(1)	-1
	Bruno(5, 3, [0, 1, 0])	2		
	Anna(5)			
			DrawCard(-1)	1
			DrawCard(0)	1
			DrawCard(1)	0
			DrawCard(2)	1
			DrawCard(-1)	0
			DrawCard(1)	-1
	Bruno(5, 4, [1, 0, 1, 0])	3		

The sample input consists of Q (= 2) plays of fortune-telling, and the number of cards Anna draws from the deck is N (= 5). In this example, in the first fortune-telling, Anna plays in the following procedure:

1. Anna draws a card from the deck. The number written on the card is 0.
2. Anna selects to put the card on the table, and puts it on the left of the sequence of cards on the table. Then, the sequence of cards becomes 0. Anna draws another card from the deck, and 1 is written on it.
3. Anna selects to discard the card. Then, Anna draws another card from the deck, and 0 is written on it.
4. Anna selects to put the card on the table, and puts the card immediately right to the 1-st leftmost card on the table. Then, the sequence of cards becomes $0, 0$. Anna draws another card from the deck, and 0 is written on it.
5. Anna selects to discard the card. Then, Anna draws another card from the deck, and 1 is written on it.

6. Anna selects to put the card on the table, and puts the card immediately right to the 1-st leftmost card on the table. Then, the sequence of cards becomes $0, 1, 0$.

Then, Bruno sees the sequence of cards: $0, 1, 0$. Based on this information, Bruno guesses that the number of cards with 1 among the cards Anna drew is 2. Since this is correct, fortune-telling is a success. The number of cards Anna put on the table is $L = 3$.

In the second fortune-telling, the number of cards Anna put on the table is $L = 4$.

Note that this sample input **does not satisfy** the constraints of the problem. The file `sample-01-in.txt`, which can be downloaded from the contest site, corresponds to Sample Input 1. The file `sample-02-in.txt`, which can be downloaded from the contest site, is a sample input that satisfies the constraints.