Rybina search (rybina)

At the Schueberfouer, the largest funfair in Luxembourg, there's a popular number guessing game. The showman, Rybina, hides a secret password of the form AB where A and B are N-digit strings. He allows you to guess the password, and after each guess, he will tell you whether your guess is too high or too low.

However, after each guess, Rybina might swap the two halves of the password, i.e. AB becomes BA (or BA becomes AB). He will also tell you if he has done so. Your next guess will then be compared against the swapped password.

It is your task to figure out the secret password using at most 64 guesses.

Implementation

You will have to submit a single `.cpp` source file.

⚠ Among this task's attachments you will find a template `rybina.cpp` with a sample implementation.

You will have to implement the following function:

C++	`void rybina_search(int N)`

- The integer N represents the number of digits in each half of the password.

Your program can call the following function, which is already defined in the grader:

C++	`GuessResult guess(long long X)`

- The integer X represents the password you have entered as a guess. It should be nonnegative and should contain at most $2N$ digits when written out in decimal (i.e. $0 \leq X < 10^{2N}$)
- If your guess is correct, your program is terminated and you receive the `accepted` verdict for the current test case.
- Otherwise the function returns a `GuessResult`, a struct with an integer field `comparison` and a boolean field `swapped`.
- The value of `comparison` is 1 when your guess was too low and -1 if your guess was too high.
- The value of `swapped` indicates whether the password has swapped after your guess was made.

Note that the grader **is adaptive** and the correct answer or the swaps for a test case are therefore not fixed beforehand.

Sample Grader

Among this task's attachments you will find a simplified version of the grader used during evaluation, which you can use to test your solutions locally. The sample grader reads data from `stdin` and calls the function `rybina_search`, using the following input format.

The input consists of two lines:

- Line 1: the integer N.

- Line 2: two integers S and T. The first, S, is the seed used to decide the random swaps; the second, T, is the target: the initial password AB written as a single integer with $0 \leq T < 10^{2N}$.

After each guess the sample grader swaps the two halves of the password at random, using the seed S. (The grader used during the official evaluation is adaptive instead.)

The sample grader reports the outcome on `stderr`: when your solution guesses the password it prints the number of guesses used, and otherwise a brief message with the reason (an invalid guess, or too many guesses).

Constraints

- $1 \leq N \leq 9$.

You can call the function `guess` at most 64 times.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 0** [**0 points**]: Sample test cases.
- **Subtask 1** [**5 points**]: $N = 1$ and $B = 0$, i.e. the second half of the initial password is `0`.
- **Subtask 2** [**5 points**]: $N = 1$.
- **Subtask 3** [**10 points**]: $N \leq 4$ and Rybina will never swap the password.
- **Subtask 4** [**20 points**]: Rybina will never swap the password.
- **Subtask 5** [**15 points**]: Rybina will always swap the password.
- **Subtask 6** [**20 points**]: $N \leq 4$
- **Subtask 7** [**25 points**]: No additional constraints.

Examples

Grader	Solution
`rybina_search(2)`	
	`guess(1000)`
`return [1, 1]`	
	`guess(1000)`
`return [-1, 0]`	
	`guess(412)`
`return [0, 0]`	

Explanation

In the first example, $N = 2$ and the two parts of the secret password are `12` and `04`. The guess 1000 is made, at this point in time the password is 1204, so the guess is too low and after the guess is made, Rybina reports that the password is swapped. The password is now 412 and the same guess of 1000 is made. This time the guess is too high and Rybina also reports that the password has not been swapped afterwards. Finally the password 412 is guessed, which turns out to be correct and the program is done.