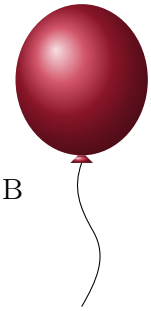
F Game: Pattern Chase



Time limit: 1.0s
Memory limit: 1024MB

This is an interactive problem.

Alice and Bob are playing a game with a binary string s and a fixed integer k. Initially, there is an empty string t. The players take turns appending a character `0` or `1` to the end of t, starting with Alice.

The interaction always continues until exactly k characters have been appended to t. Alice wins if and only if the final string t contains s as a contiguous substring. Otherwise, Bob wins.

You may choose to play as either Alice or Bob. Your goal is to win the game against the jury.

Interaction

Each test run contains multiple test cases. You should first read a line with an integer T ($1 \le T \le 100$), representing the number of test cases.

For each test case, you begin the interaction by reading a binary string s and an integer k in a single line ($1 \le |s| \le k \le 100$), denoting the number of rounds and parameters of the game. Afterwards, output one word: `Alice` if you choose to play as Alice, or `Bob` if you choose to play as Bob. After that, the game starts from the empty string. Alice makes the first move. Whenever it is your turn, output one character, either `0` or `1`. Whenever it is the jury's turn, read one character, either `0` or `1`. **The game ends when the current string has length k.**

If there are multiple choices of role or moves that can make you win, you may output any of them, as long as your interaction follows the protocol and wins the game.

After every output operation you must flush the output buffer. For example, in C++ you may use `cout << value << endl;` or `cout.flush();`.

If you output an invalid token, make a move after the game has ended, fail to flush, or lose the game, you will receive Wrong Answer.

Samples

Sample input 1	Sample output 1
1 01 3 (receiving participant's output) (receiving participant's output) 0 (receiving participant's output)	(receiving jury's output) (receiving jury's output) Alice 0 (receiving jury's output) 1

Explanation of sample 1.
The lines of the form `(receiving ... output)` are placeholders used only to align the judge stream and the contestant stream in the sample. In a real test case, the jury will not output these placeholder lines, and the participant should neither read nor print them.

The bullet points below describe the interaction shown in the sample.

- The jury sends `1`, so there is one test case, then sends `01 3`, so $s = \texttt{01}$ and $k = 3$.
- The contestant chooses `Alice` and appends `0`, so the current string becomes `0`.
- The jury, playing Bob, appends `0`; now the current string is `00`.
- Alice appends `1`. The final string is `001`, which contains `01`, so Alice wins.

Local interaction tool: The attached `attachments/local_interactive.py` can be used for local testing with random jury moves, but passing it does not guarantee passing the official judging data.