Problem J
Base Hi-Lo Game
Time limit: 2 seconds

For this problem, you will write a program that plays a guessing game to figure out a number of up to 64 digits in a specified base (2–62). Digits are: `0-9, A-Z, a-z` in that order.

This is an interactive problem. All lines sent by your program must be **newline terminated and flushed out** (you may have to disable buffering). All responses received by your program will be **newline terminated lines**.

Your program will make a guess of what the number might be, and the judging system will respond with a string giving you information about each digit in your guess.

There is a limit on the number of guesses you are allowed to make for each test case, after which point, if you have not guessed the number, your program will be terminated and given a `WA` (wrong answer) judgment. The number of guesses you are allowed is $floor(log_2(base)) + 2$. To make it interesting, it's possible that the judging system may try to *cheat* you and give you faulty information back. If the judging system does decide to cheat, it will cheat only once, for a single digit.

If the number you guess is correct, the judging system will *always* respond with the string "`correct`" (with no quotes).

Sample Interaction

When your program starts, it should read two space separated decimal integer values: the base B of all the test cases, $(2 \leq B \leq 62)$, and the number of test cases, N $(1 \leq N \leq 100)$.

Your program will then read a decimal value, which is the number of digits D in the first test case, $(1 \leq D \leq 64)$. Your program will then respond with a line consisting of a D digit, base B value which is your first guess. You will then read back a line containing a string from the judging system. The action your program takes now depends on the string you read from the judging system.

- Case 1: If the value of the string is "`correct`", you have successfully guessed the number. If there are more test cases, your program should go back and wait for the length D of the next test case. If there are no more test cases, your program should exit.
- Case 2: The string is a D character string consisting of characters in the set: `[+,-,=]`. Each character is an indication as to the accuracy of the digit in the corresponding position of your guess. A + means the digit in that position is too small. A − means the digit in that position is too big. An = means the digit is correct. You will then make a new guess based on the information you got back. Your program will repeat this process until you receive a "`correct`" response, or the judge terminates your program because it took too many guesses.

If at any point during the guessing dialog you detect that the judging system is cheating, you should send back a line with the word "`cheater`" on it (without quotes). If the judging system was, in fact, cheating, you will receive a "`correct`" (without quotes) response back. If the judging system was not cheating, your program will be terminated and it will receive a `WA` response. An example of cheating might be if the judging system responds in different ways to the same guess. Another example might be contradictory information: a judging system response to an earlier guess conflicts with the response of a later guess. The judging system may cheat only once per test case.

For **Sample Interaction 1**, the correct value is `GNY23`. For **Sample Interaction 2**, the correct values are `555` and `9`.

Sample Interaction 1

Read	Write

`36 1`
`5`	

	`00000`
`+++++`	
	`55555`
`+++--`	
	`AAA33`
`+++-=`	
	`GGG13`
`=+++=`	
	`GNN23`
`==+==`	
	`GNT23`
`(no response - too many guesses - WA)`	

Sample Interaction 2

Read	Write

`10 2`
`3`	

	`543`
`=++`	
	`554`
`==+`	
	`555`

`correct`
`1`	

	`2`
`+`	
	`5`
`+`	
	`7`
`+`	

```
8
```

```
(no response - too many guesses - WA)
```