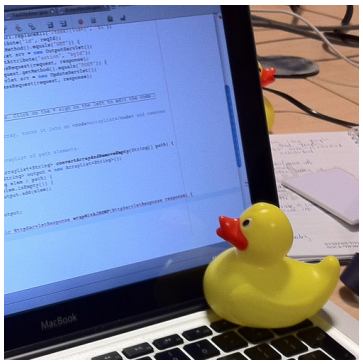
D Ducky Debugging



CC BY-SA 3.0 By Tom Morris on Wikipedia

You don't know anything about programming, but nonetheless you and your friend Bob joined the BAPC. Bob just submitted a solution to a problem, but sadly his submission was not accepted. You will help him figure out what mistake he made. In order to do so, he will explain what every part of the program should do according to him. He does not want you to interrupt him while he is explaining, but whenever he is unsure of something, he will ask for confirmation. Bob is actually a very good coder, so whenever he asks for confirmation, he is simply right. You just need to react by nodding and saying "Quack!". As long as you don't interrupt him at the wrong time, and as long as you say "Quack!" at the right times, he will eventually find his mistake and he will shout "I quacked the code!".

Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

The interactor sends one line of text at a time, each line being one sentence of Bob's monologue. You need to reply as follows:

- If the line ends with a question mark ("`?`"), your reply must be the string "Quack!".
- If the line ends with a period ("`.`"), your reply must be the string "*Nod*".
- If the line is exactly "I quacked the code!", your submission should exit.
  Reading more input will result in a time limit exceeded and printing more output will result in a wrong answer.

Bob's monologue will be no more than 100 sentences, with at most 10 000 characters in total. It is guaranteed that every sentence, except the last one, ends with a question mark ("`?`") or a period ("`.`").

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

Read	**Sample Interaction 1**	**Write**
`In the loop, I read a line of input.`		
		`*Nod*`
`The method std::getline(std::cin, line); reads until a newline, right?`		

```
                                                    Quack!
Then I branch according to the last
character of the line of input.
                                                    *Nod*
When the character is an exclamation
mark, I terminate the program.
                                                    *Nod*
When it is a question mark, I need
to print "Quack!" right?
                                                    Quack!
In any other case, I simply print
"*Nod*".
                                                    *Nod*
Wait, to flush the output stream
I need to use std::endl instead of
'\n' right?
                                                    Quack!
I quacked the code!
```

Read **Sample Interaction 2** **Write**

```
My program seems to always exit
after reading one line.
                                                    *Nod*
Shouldn't it only halt when a line
ends with an exclamation mark?
                                                    Quack!
After reading a line, I check the
last character using a switch.
                                                    *Nod*
When the character is a period, I
print "*Nod*".
                                                    *Nod*
When the character is a question
mark, I print "Quack!".
                                                    *Nod*
Wait, both get printed before the
program exits?
                                                    Quack!
I guess I forgot to add break; here
and there...
                                                    *Nod*
I quacked the code!
```