SWITCH TO EDITOR

Problem

Last year, a research consortium had some trouble with a distributed database system that sometimes lost pieces of the data. You do not need to read or understand that problem in order to solve this one!

The consortium has decided that distributed systems are too complicated, so they are storing **B** bits of important information in a single array on one awesome machine. As an additional layer of security, they have made it difficult to obtain the information quickly; the user must query for a bit position between 1 and **B**, and then they receive that bit of the stored array as a response.

Unfortunately, this ultra-modern machine is subject to random quantum fluctuations! Specifically, after every 1st, 11th, 21st, 31st... etc. query is sent, *but before the response is given*, quantum fluctuation causes exactly one of the following four effects, with equal probability:

- 25% of the time, the array is complemented: every `0` becomes a `1`, and vice versa.
- 25% of the time, the array is reversed: the first bit swaps with the last bit, the second bit swaps with the second-to-last bit, and so on.
- 25% of the time, both of the things above (complementation and reversal) happen to the array. (Notice that the order in which they happen does not matter.)
- 25% of the time, nothing happens to the array.

Moreover, there is no indication of what effect the quantum fluctuation has had each time. The consortium is now concerned, and it has hired you to get its precious data back, in whatever form it is in! Can you find the entire array, such that your answer is accurate *as of the time that you give it*? Answering does not count as a query, so if you answer after your 30th query, for example, the array will be the same as it was after your 21st through 30th queries.

Input and output

This is an interactive problem. You should make sure you have read the information in the Interactive Problems section of our FAQ.

Initially, your program should read a single line containing two integers **T** and **B**: the number of test cases and the number of bits in the array, respectively. Note that **B** is the same for every test case.

Then, you need to process **T** test cases. In each case, the judge begins with a predetermined **B**-bit array; note that this array can vary from test case to test case, and is not necessarily chosen at random. Then, you may make up to 150 queries of the following form:

- Your program outputs one line containing a single integer P between 1 and **B**, inclusive, indicating which position in the array you wish to look at.
- If the number of queries you have made so far ends with a 1, the judge chooses one of the four possibilities described above (complementation, reversal, complementation + reversal, or nothing), uniformly at random and independently of all other choices, and alters the stored array accordingly. (Notice that this will happen on the very first query you make.)
- The judge responds with one line containing a single character `0` or `1`, the value it currently has stored at bit position P, or `N` if you provided a malformed line (e.g., an invalid position).