Jeopardized Betting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

This is an interactive problem.

In knockout competitions, two teams play matches between each other until one of the teams wins N of them. At the same time, there can be no tied matches, so each match ends with the victory of one of the teams.

Your favourite team is playing in one of these competitions and you want to bet on this team winning the series. The bet should work like this: if your team wins the series, double the bet is returned, and if it loses, nothing is returned.

However, the bookmaker you are using wants you to place as many bets as possible. Instead of giving the opportunity to bet on the outcome of the entire series of games, you are offered only to bet on each game separately.

But you didn't give up and accepted the bookmaker's challenge. You want to bet on each match in such a way that by the end of the series the amount of money you have either doubles (if your team wins), or is spent on bets and you have nothing left (in case of defeat of your team).

Interaction Protocol

First, your program must read an integer N — the number of matches required to win the series ($1 \le N \le 30$).

Initially your money is equal to $2^{2 \cdot N}$.

Then your program must bet on the first game of the series. To bet, you shall print a line containing one integer b — the amount of money used to bet. The value of b shall be strictly positive and should not exceed the amount of money you currently have.

The betting process then proceeds as follows.

After placing a bet, the jury program tells you the result of the match:

- `Won` — means that your team won and you get double your bet.
- `Lost` — means that the bet did not play and the money will not be returned to you.

If after the next match one of the teams scores N wins, then the series of games ends and your program should exit.

If the series continues, then you must bet on the next game.

It is guaranteed that there always exists a sequence of bets that will solve the task.

Make sure you print the newline character and flush the output stream buffer (`flush` language command) after each printed bet. Otherwise, the solution may get the idleness limit exceeded verdict.

Example

standard input	standard output
3	
	40
Lost	
	20
Won	
	40
Won	
	44
Won	