Two-Way Communication

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	1024 megabytes

This is a communication problem.

Two programs, Alice and Bob, play the following cooperative game.

You are given integers A and B with $0 \leq A, B < 2^{64}$. Initially, only Alice knows A, and only Bob knows B. Let $C := \min(A, B)$. The goal is for **both** Alice and Bob to be able to output C.
To exchange information, Alice and Bob may call the following functions:

- `send(x)`: Send a 1-bit value x to the other program. This call does not return until the other program calls `receive()`.
- `receive()`: Receive a 1-bit value from the other program and return it. This call does not return until the other program calls `send()`.

The total number of calls to `send()` made by Alice and Bob combined must not exceed 76. The same holds for `receive()`.
After communication is finished, each program must call the following function **exactly once**:

- `answer(x)`: Declare that the value of C is x. Immediately after calling this function, the program must terminate.

The game is successful if both Alice and Bob answer C correctly.

Write two programs, Alice and Bob, that always succeed in the game.

Interaction Protocol

This is an **interactive communication problem** (your programs and the judge communicate via standard input/output). Your program acting as Alice and your program acting as Bob will play the game through the judge.

Program Start

Your program first receives input in the following format:

Player
X

Here, *Player* is the string `Alice` or `Bob`, and X is an integer with $0 \leq X < 2^{64}$.

- If *Player* is `Alice`, then $A = X$. From this point on, **your program must behave as Alice**.
- If *Player* is `Bob`, then $B = X$. From this point on, **your program must behave as Bob**.

After that, call `send`, `receive`, and finally `answer` as described below.

send

To call `send(x)`, output:

```
send x
```

Here, x must be 0 or 1.

This call does not return until the other program calls `receive()`. You will be judged as **Wrong Answer** if any of the following happens:

- the other program terminates while you are waiting,
- the other program also calls `send()` while you are waiting,
- the total number of calls to `send()` made by Alice and Bob exceeds 76.

receive

To call `receive()`, output:

```
receive
```

Then read the received bit x from standard input:

```
x
```

This call does not return until the other program calls `send()`. You will be judged as **Wrong Answer** if any of the following happens:

- the other program terminates while you are waiting,
- the other program also calls `receive()` while you are waiting,
- the total number of calls to `receive()` made by Alice and Bob exceeds 76.

If you are judged as Wrong Answer during the interaction, the input provided will be `-1`. **If you read `-1`, terminate your program immediately**, or you may receive Time Limit Exceeded.

answer

To call `answer(x)`, output:

```
answer x
```

Here, x must be equal to $\min(A, B)$. **After calling this function, terminate your program immediately.**

Note

- After each output, your program must flush standard output; Otherwise, you will receive Time Limit Exceeded.
- The judge program, your program acting as Alice, and your program acting as Bob run simultaneously. **The time and memory limits are measured as the sum of all of them**, so leave enough margin for both time and memory usage.
    - The judge program consumes up to about 50 ms and 5 MiB.
- You will handle integers in the range 0 to $2^{64} - 1$. Be careful about overflow.

Sample Interaction

Alice's Input	Alice's Output	Bob's Input	Bob's Output
Alice 31		Bob 25	
	send 0		receive
		0	
	receive		send 1
1			
	send 1		receive
		1	
	answer 25		answer 25