

Boring Game

Alice and her little brother, Bob are playing a number guessing game.

Bob has selected a (hidden) integer S .

Alice can ask questions about the hidden number, which are of the following form: "Is the hidden number at least x ?" Bob answers her questions with "Yes" or "No". Unfortunately, after $K \geq 1$ questions, Bob gets bored of the game, and from then on, he will give false answers to all questions.

That is, Bob:

- Answers "Yes" to the first K questions if and only if $x \leq S$, and
- After the K -th question, he answers "Yes" if and only if $S < x$.

Note that Bob always answers correctly to the first question and Alice does not know the value of K .

Your task is to devise and implement a questioning strategy for Alice to identify the hidden number. Your score is based on the number of questions asked - the fewer questions, the better the score.

Implementation Details

You should implement the following procedure.

```
long long play_game()
```

- This procedure is called **at most T times** for each test case.

The procedure should find and return the hidden number S by making calls to the following procedure to ask questions about the hidden number:

```
bool ask(long long x)
```

- x : the number specifying a question of Alice.
- The value of x must be between 1 and 10^{18} , inclusive.
- The procedure returns a boolean value representing Bob's answer.

- The procedure can be called at most 150 times in each call to `play_game`.

The behaviour of the grader is **adaptive**, meaning that in certain tests, the values of S and K are not fixed before `play_game` is called. It is guaranteed that there exists at least one (S, K) pair for which the grader's answers are consistent.

Constraints

- $1 \leq T \leq 1000$
- $1 \leq S \leq 10^{18}$
- $1 \leq K \leq 150$

Scoring

If your solution does not adhere to the implementation details described above, or if the value returned by `play_game` is incorrect for even a single call, your solution will receive a score of 0.

Otherwise, let C be the maximum number of questions your solution asks across all calls to `play_game`. Your score is calculated according to the following table:

Condition	Points
$132 < C \leq 150$	$20 \cdot \left(\frac{150-C}{18}\right)^2$
$78 < C \leq 132$	$20 + 20 \cdot \left(\frac{132-C}{54}\right)^2$
$72 < C \leq 78$	$40 + 30 \cdot \left(\frac{78-C}{6}\right)^2$
$67 < C \leq 72$	$70 + 30 \cdot \left(\frac{72-C}{5}\right)^2$
$C \leq 67$	100

Example

Consider a scenario where the hidden integer is $S = 2$, and $K = 1$. The game begins with the call:

```
play_game()
```

Suppose that `play_game` first calls `ask(2)`. As $2 \leq S = 2$ and Bob is telling the truth at this point, the call returns `true`.

Suppose `play_game` calls `ask(2)` again. Although the condition $2 \leq S = 2$ still holds, but Bob is now lying (having already told the truth $K = 1$ time), so the call returns `false`. Receiving contradictory answers to the same question reveals that Bob will lie on all subsequent responses.

Now suppose `play_game` calls `ask(3)`. Since $3 \leq S = 2$ is false, and Bob is lying, the call returns `true`. At this point, we can deduce that $2 \leq S < 3$, which implies $S = 2$.

Therefore, the procedure should return 2.

Sample Grader

The sample grader is **not adaptive**. It processes T scenarios, and in each case reads fixed values of S and K from the input and answers questions accordingly.

Input format:

```
T
S[0]  K[0]
S[1]  K[1]
...
S[T-1] K[T-1]
```

Here, $S[i]$ and $K[i]$ ($0 \leq i < T$) specify the hidden parameters for each call to `play_game`.

Output format:

```
G[0]  C[0]
G[1]  C[1]
...
G[T-1] C[T-1]
```

Here $G[i]$ ($0 \leq i < T$) is the number returned by `play_game` when the hidden parameters are $S[i]$ and $K[i]$, and $C[i]$ is the number of questions asked during that call.