El gran premio

El gran premio es un famoso programa de televisión. Eres el afortunado ganador que ha llegado a la última ronda. Te encuentras frente a una fila de n cajas, numeradas del 0 al $n-1$ de izquierda a derecha. Cada caja contiene un premio que no se puede ver hasta que la caja se abra. Existen $v \geq 2$ *tipos* de premio diferentes. Los tipos de premios están numerados de 1 a v en orden *decreciente* respecto a su valor.

El premio de tipo 1 es el más valioso: un diamante. Existe exactamente un diamante dentro de las cajas. El premio de tipo v es el menos valioso: un dulce. Para hacer el juego más interesante, el número de premios baratos es mucho mayor que el número de premios valiosos. Específicamente, para todo t tal que $2 \leq t \leq v$ sabemos lo siguiente: si hay k premios del tipo $t-1, entonces existen *estrictamente más* de k^2 premios del tipo t.

Tu objetivo es ganar el diamante. Al final del juego deberás abrir una caja y ganarás el premio que contiene. Antes de abrir la caja, puedes hacerle preguntas a Rambod, el conductor del programa. Para cada pregunta escoges una caja i. Rambond te responderá con un arreglo a de 2 enteros que significan lo siguiente:

- De todas las cajas a la izquierda de i hay exactamente $a[0]$ cajas que contienen un premio más valioso que el premio de la caja i.
- De todas las cajas a la derecha de i hay exactamente $a[1]$ cajas que contienen un premio más valioso que el premio de la caja i.

Por ejemplo, supón que $n = 8$. Para tu pregunta escoges la caja $i = 2$. Rambond te responde que $a = [1, 2]$. Esto significa:

- Exactamente una de las cajas 0 ó 1 contienen un premio más valioso que la caja 2.
- Exactamente dos de las cajas $3, 4, \ldots, 7$ contienen un premio más valioso que la caja 2.

Tu tarea es encontrar la caja con el diamante haciendo tan pocas preguntas como puedas.

Detalles de implementación

Debes implementar el siguiente procedimiento:

```
int find_best(int n)
```

- n: número de cajas.
- Este procedimiento debe regresar el índice de la caja que contiene el diamante, esto es, el único entero d ($0 \leq d \leq n-1$) tal que la caja d es un premio de tipo 1.

Tu procedimiento puede hacer llamadas al siguiente procedimiento:

```
int[] ask(int i)
```

- i: índice de la caja por la que quieres preguntar, el valor de i debe estar entre 0 y $n-1$, inclusive.
- Este procedimiento devuelve un arreglo a con dos elementos. $a[0]$ es el número de premios más valiosos en las cajas a la izquierda de la caja i y $a[1]$ es el número de premios más valiosos en las cajas a la derecha de la caja i.

Ejemplo

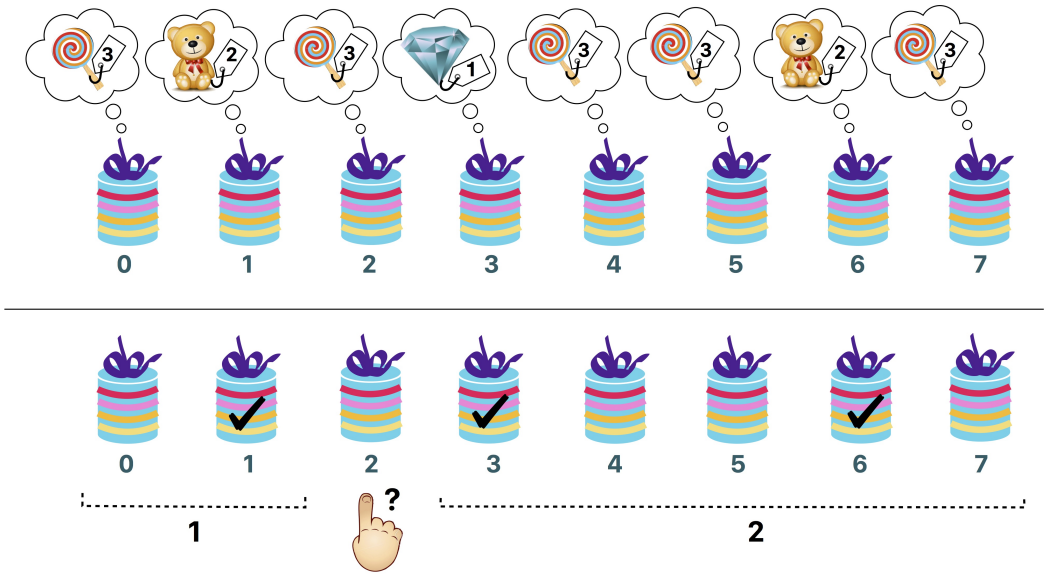
El evaluador hace la siguiente llamada:

```
find_best(8)
```

Existen $n = 8$ cajas. Supón que los premios son $[3, 2, 3, 1, 3, 3, 2, 3]$. Todas las posibles llamadas al procedimiento `ask` y sus respuestas correspondientes se listan a continuación.

- `ask(0)` regresa $[0, 3]$
- `ask(1)` regresa $[0, 1]$
- `ask(2)` regresa $[1, 2]$
- `ask(3)` regresa $[0, 0]$
- `ask(4)` regresa $[2, 1]$
- `ask(5)` regresa $[2, 1]$
- `ask(6)` regresa $[1, 0]$
- `ask(7)` regresa $[3, 0]$

En este ejemplo el diamante está en la caja 3 por lo que el procedimiento `find_best` debe regresar 3.



La figura anterior ilustra este ejemplo. La parte de arriba muestra los tipo de premio de cada caja. La parte inferior ilustra la pregunta `ask(2)`. Las cajas marcadas contienen premios mas valiosos que la caja con índice 2.

Restricciones

- $3 \leq n \leq 200\,000$.
- El tipo de premio en cada caja tiene indice entre 1 y v, inclusive.
- Existe exactamente un premio de tipo 1.
- Para todo $2 \leq t \leq v$, si existen k premios de tipo $t-1$, entonces existen *estrictamente* más de k^2 premios de tipo t.

Subtareas y puntuación

En algunos casos de prueba el comportamiento del evaluador es adaptativo. Esto significa que en estos casos de prueba el evaluador no tiene una secuencia fija de premios. es decir, las respuestas que da pueden depender de las preguntas que realiza tu solución. Se garantiza que las respuestas del evaluador se darán de tal forma que exista al menos una secuencia de premios consistente con todas las respuestas presentadas hasta ese momento.

1. (20 puntos) Existe exactamente 1 diamante y $n-1$ dulces (osea, $v=2$). Puedes llamar el procedimiento `ask` a lo más $10,000$ veces.
2. (80 puntos) Sin restricciones adicionales.

En la subtarea 2 puedes obtener puntos parciales. Dentro de todos los casos de esta subtarea, definimos q como el máximo número de llamadas al procedimiento `ask`. Tu puntaje será calculado de acuerdo a la siguiente tabla:

Preguntas	Puntaje
$10\,000 < q$	0 ('Wrong Answer' en el CMS)
$6000 < q \leq 10\,000$	70
$5000 < q \leq 6000$	$80 - (q - 5000)/100$
$q \leq 5000$	80

Evaluador de ejemplo

El evaluador de ejemplo no es adaptativo. Solo lee y usa un arreglo fijo p de tipos de premios. Para todo $0 \leq b \leq n - 1$, el tipo de premio en la caja b es dado por $p[b]$. El evaluador de ejemplo espera la entrada en el siguiente formato:

- linea 1: n
- linea 2: $p[0]$ $p[1]$ $\ldots$ $p[n-1]$

El evaluador de ejemplo imprime una sola línea con el valor regresado por `find_best` y el número de llamadas al procedimiento `ask`.