El Gran Premio

El Gran Premio es un famoso programa de televisión. Eres el afortunado participante que ha logrado alcanzar la ronda final. Te encuentras parado delante de una fila de n cajas, numeradas desde 0 hasta $n-1$, de izquierda a derecha. Cada caja contiene un premio que no es revelado hasta que no se abre la caja. Hay $v \geq 2$ *tipos* de premios diferentes. Los tipos se numeran desde 1 hasta v, en orden *decreciente* de valor.

El premio de tipo 1 es el más caro de todos: un diamante. Hay exactamente un diamante en las cajas. El premio de tipo v es el más barato de todos: un chupetín. Para que el juego sea más emocionante, hay muchos más premios entre los más baratos que entre los más caros. Más precisamente, para cada t tal que $2 \leq t \leq v$ sabemos lo siguiente: si hay k premios de tipo $t-1$, hay *estrictamente más* de k^2 premios de tipo t.

Tu objetivo es ganar el diamante. Al finalizar el juego, deberás abrir una caja y ganarás el premio que esta contenga. Antes de elegir qué caja abrir, tienes la oportunidad de hacerle algunas preguntas a Rambod, el anfitrión del programa. Para cada pregunta, deberás elegir una cierta caja i. En respuesta a la pregunta, Rambod te dará un arreglo a con dos enteros. Su significado es el siguiente:

- Entre las cajas que se encuentran a la izquierda de la caja i, hay exactamente $a[0]$ cajas que contienen un premio más caro que el que hay en la caja i.
- Entre las cajas que se encuentran a la derecha de la caja i, hay exactamente $a[1]$ que contienen un premio más caro que el que hay en la caja i.

Por ejemplo, supongamos que $n = 8$ y que decides preguntar por la caja $i = 2$. La respuesta de Rambod será $a = [1, 2]$. Esta respuesta significa que:

- Exactamente una de las cajas 0 y 1 contiene un premio más caro que el que hay en la caja 2.
- Exactamente dos de las cajas $3, 4, \ldots, 7$ contienen un premio más caro que el que hay en la caja 2.

Tu tarea consiste en identificar la caja que contiene el diamante, realizando pocas preguntas en total.

Detalles de implementación

Debes implementar la siguiente función:

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

Esta función es llamada exactamente una vez por el evaluador.

- n: cantidad de cajas.
- Esta función debe retornar el número de la caja que contiene el diamante, es decir, el único entero d ($0 \leq d \leq n-1$) tal que la caja d contiene un premio de tipo 1.

La función anterior puede llamar a la siguiente:

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

- i: número de caja por la cual se quiere preguntar. El valor de i debe estar entre 0 y $n-1$, inclusive.
- Esta función devuelve el arreglo a con 2 elementos, de forma que $a[0]$ sea la cantidad de premios más caros en cajas a la izquierda de la caja i, y $a[1]$ sea la cantidad de premios más caros en cajas a la derecha de la caja i.

Ejemplo

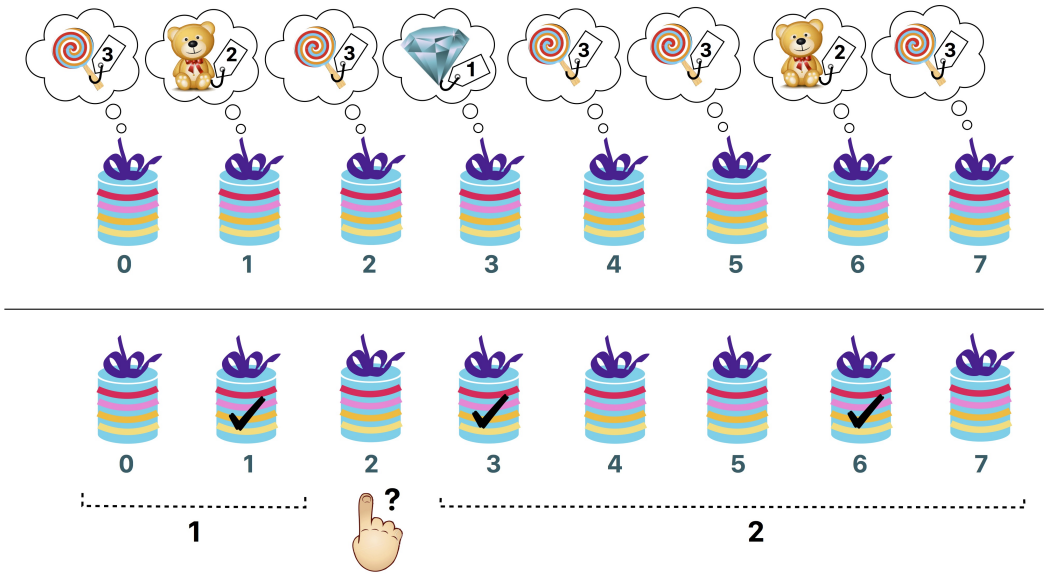
El evaluador hace la siguiente llamada:

```
find_best(8)
```

Hay $n = 8$ cajas. Supongamos que los tipos de premios son $[3, 2, 3, 1, 3, 3, 2, 3]$. Se muestran a continuación todas las posibles llamadas a la función `ask`, junto con sus correspondientes resultados.

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

En este ejemplo, el diamante se encuentra en la caja 3. Por lo tanto la función `find_best` debe retornar 3.



La figura se corresponde con este ejemplo. La parte superior muestra los tipos de los premios que hay en cada caja. La parte inferior muestra la consulta `ask(2)`. Las cajas marcadas en la figura contienen premios más caros que el que se encuentra en la caja 2.

Restricciones

- $3 \leq n \leq 200\,000$.
- El tipo de premio que hay en cada caja se encuentra entre 1 y v, inclusive.
- Hay exactamente un premio de tipo 1.
- Para cada $2 \leq t \leq v$, si hay k premios de tipo $t-1$, hay *estrictamente* más de k^2 premios de tipo t.

Subtareas y puntaje

En algunos casos de prueba, el comportamiento del evaluador es adaptativo. Esto quiere decir que en estos casos de prueba, el evaluador no tiene una secuencia de premios fija. En lugar de esto, las respuestas dadas por el evaluador podrían depender de las preguntas realizadas por tu solución. Se garantiza que el evaluador responderá de forma tal que luego de cada respuesta, exista al menos una secuencia de premios consistente con todas las respuestas dadas hasta el momento.

1. (20 puntos) Hay exactamente 1 diamante y $n-1$ chupetines (o sea, $v = 2$). Se puede llamar a la función `ask` como máximo $10\,000$ veces.
2. (80 puntos) Sin restricciones adicionales.

En la subtarea 2, es posible obtener un puntaje parcial. Sea q la cantidad máxima de llamadas a la función `ask` entre todos los casos de prueba de esta subtarea. Tu puntaje para esta subtarea se calcula de acuerdo a la siguiente tabla:

Questions	Score
$10\,000 < q$	0 (en CMS será 'Wrong Answer' o "Respuesta Incorrecta")
$6000 < q \leq 10\,000$	70
$5000 < q \leq 6000$	$80 - (q - 5000)/100$
$q \leq 5000$	80

Evaluador de ejemplo

El evaluador de ejemplo no es adaptativo. En lugar de ello, simplemente lee y utiliza un arreglo prefijado p de tipos de premios. Para cada $0 \leq b \leq n - 1$, el tipo del premio en la caja b viene dado por $p[b]$. El evaluador de ejemplo lee la entrada con el siguiente formato:

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

El evaluador de ejemplo imprime una única línea con el resultado de `find_best` y la cantidad de llamadas realizadas a la función `ask`.