# El Premio Mayor

El premio mayor es un espectaculo de juego de la TV. Usted es un concusante afortunado que ha avanzado a la ronda final. Usted esta parado al frente de una fila de $n$ cajas, etiquetadas desde $0$ hasta $n-1$ de izquierda a derecha. Cada caja contiene un premio que no puede verse hasta que la caja sea abierta. Hay $v \geq 2$ diferentes *tipos* de premios. Los tipos son numerados desde $1$ hasta $v$ en orden *decreciente* respecto a su valor.

El premio de tipo $1$ es el mas caro: un diamante. Hay exactamente un diamante en las cajas. El premio de tipo $v$ es el mas barato: un chupete. Para hacer el juego mas emocionante, el numero de premios baratos es mucho mas grande que el numero de premios caros. Especificamente, para todo $t$ tal que $2 \leq t \leq v$ conocemos que: si hay $k$ premios de tipo $t-1$, hay *estrictamente mas* de $k^2$ premios del tipo $t$.

Su objetivo es ganar el diamante. Al final del juego usted tendra que abrir una caja y recibir el premio que esta contiene. Antes de abrir la caja, puedes hacerle preguntas a Rambod, el conductor del programa. Por cada pregunta usted selecciona alguna caja $i$. Como respuesta, Rambod le dara un arreglo $a$ que contiene dos enteros. Su significado es como sigue:

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

Por ejemplo, suponga que $n = 8$. Para su pregunta usted selecciona la caja $i = 2$. Como la respuesta de Rambod dice que $a = [1, 2]$. El significado de esta respuesta es:

- Exactamente una de las cajas $0$ o $1$ contiene un premio mas caro que el de la caja $2$.
- Exactemente dos de las cajas $3, 4, \ldots, 7$ contiene un premio mas caro que el de la caja $2$.

Su tarea es encontrar la caja que contiene el diamante, haciendo el menor numero de preguntas.

## Detalles de la Implementacion

Usted debe implementar la siguiente funcion:

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

- El evaluador llama a la funcion exactamente una vez.
- $n$: es el numero de cajas.

- Esta funcion debe retornar la etiqueta de la caja que contiene el diamante, es decir, el entero unico $d$ ($0 \leq d \leq n-1$) tal que la caja $d$ contiene el premio de tipo 1.

La funcion anterior puede llamar a la siguiente funcion:

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

etiqueta de la caja que usted selecciona para preguntar sobre ella. El valor de i debe estar entre 0 y n-1, inclusive

- $i$: es la etiqueta de la caja que usted selecciona para preguntar sobre ella. El valor de $i$ debe estar entre $0$ y $n-1$, .
- Esta funcion retorna el arreglo $a$ con 2 elementos. Aqui, $a[0]$ es el numero de premios mas caros en las cajas a la izquierda de la caja $i$ y $a[1]$ es el numero de premios mas caros en las cajas a la derecha de la caja $i$.

## Ejemplo

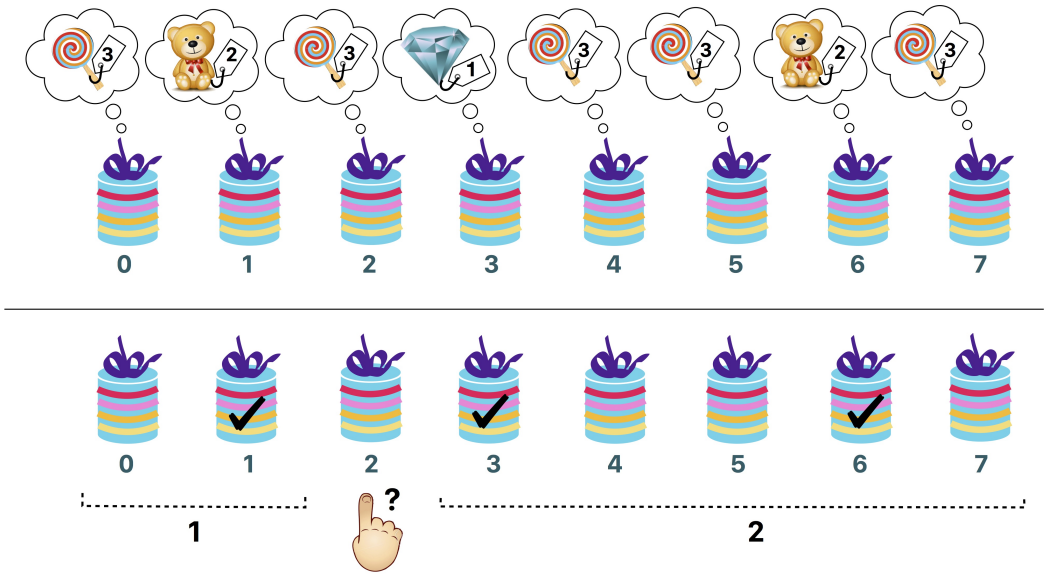
El evaluador realiza la siguiente llamada a la funcion:

```
find_best(8)
```

Hay $n = 8$ cajas. Suponga que los tipos de premios son de $[3, 2, 3, 1, 3, 3, 2, 3]$. Todas las posibles llamadas a la función `ask` y los correspondientes valores de retorno se muestran a continuacion.

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

En este ejemplo, el diamante esta en la caja $3$. Asi que la funcion `find_best` debe retornar $3$.




La figura anterior muestra este ejemplo. La parte de arriba muestra los valores de los premios en cada caja. La parte de abajo muestra la pregunta `ask(2)`. Las cajas marcadas contienen un premio mas caro que el de la caja $2$.

## Restricciones

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

## Subtareas y puntuacion

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. En lugar de esto, las respuestas que se dan por el evaluador pueden depender de una pregunta de su solucion. Se garantiza que el evaluador responda de tal forma que despues de cada respuesta exista al menos una secuencia de premios consistentes con todas las respuestas dadas hasta aqui.

1. (20 points) Hay exactamente $1$ diamante y $n-1$ chupetes (por lo tanto, $v=2$). Usted puede llamar a la funcion `ask` a lo mas $10\,000$ veces.
2. (80 points) Ninguna restriccion adicional.

En la subtarea 2 puede obtener una puntuacion parcial. Sea $q$ el maximo numero de llamadas a la funcion `ask` entre todos los casos de prueba en esta subtarea. Entonces, su puntuacion para esta subtarea se calcula de acuerdo a las siguiente tabla

| Preguntas | Puntuacion |
|---|---|
| $10\,000 < q$ | $0$ (reported in CMS as 'Wrong Answer') |
| $6000 < q \leq 10\,000$ | $70$ |
| $5000 < q \leq 6000$ | $80 - (q - 5000)/100$ |
| $q \leq 5000$ | $80$ |

## Evaluador de ejemplo

El evaluador de ejemplo no es adaptativo. En lugar de esto, solo lee y utiliza un arreglo fijo $p$ de tipo de premios. Para todo $0 \leq b \leq n - 1$, el tipo de premio de la caja $b$ se da como $p[b]$. 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 linea que contiene el valor retornado por `find_best` y el numero de llamadas a la funcion `ask`.