D. Census (census)

Límite de tiempo: 1 segundos
Límite de memoria: 128 MiB

Un dato poco conocido sobre Cesenatico es que alberga una sociedad secreta de N informáticas. Esta sociedad es muy secreta; ninguna integrante conoce la identidad de ninguna otra.

La única comunicación entre las integrantes es indirecta, a través de números garabateados con tiza en diferentes ubicaciones por toda la ciudad. Cada 100 años, la sociedad realiza un censo para contar a sus integrantes. Una vez completado el censo, cada integrante debe conocer el número total de integrantes en la sociedad.

El censo tiene lugar durante varios días. Cada día, cada integrante que aún participe en el proceso elegirá y realizará exactamente una acción: **leer**, **escribir** o **parar** de participar en él.

- Si una integrante elige **leer**, elige una ubicación P. Durante el día, visita la ubicación P y lee el número que está escrito allí.
- Si una integrante elige **escribir**, elige una ubicación P y un número V. Al final de la tarde, visita la ubicación P y cambia el número que estaba escrito allí por V. Como ya está oscuro, no puede leer el número anterior antes de escribir el nuevo.
- Si una integrante elige **parar**, ya no realizará ninguna acción en los días siguientes.

Si una integrante ve a otra escribir un número, podría conocer su identidad. Por lo tanto, está estrictamente prohibido que dos o más integrantes elijan escribir en la misma ubicación el mismo día. (No existe tal restricción para leer, ya que puede hacerse discretamente).

Si una o más integrantes leen desde una ubicación donde otra integrante quiere escribir el mismo día, todas las lecturas ocurren antes de la escritura.

¿Cómo debería la sociedad planificar su proceso de censo para minimizar el número de días hasta que todas aprendan el recuento correcto de integrantes?

Implementación

⇨ Este es un problema interactivo, en el que se ejecutarán simultáneamente un número desconocido de instancias ($1 \leq N \leq 100$) de tu programa. Cada instancia simula a una integrante de la sociedad.

Hay 10^{18} ubicaciones. El número P de una ubicación debe satisfacer $0 \leq P < 10^{18}$. Inicialmente, el valor escrito en todas las ubicaciones es $V = 0$.

El nuevo valor V escrito en una ubicación debe ser siempre un entero tal que $0 \leq V \leq 10^9$. En la mayoría de las subtareas, V solo puede ser 0 o 1. Consulta la sección de Puntuación para más detalles.

Cuando una instancia de tu programa comienza, primero debe leer una línea con dos enteros, I y M ($0 \leq I \leq M - 1$): el ID único de la integrante de la sociedad representada por esta instancia y el número total de IDs posibles. Dentro de cada caso de prueba, todas las instancias recibirán el mismo valor M y valores I distintos. Ten en cuenta que puede haber IDs que no estén asignados a ninguna integrante.

Luego, para cada día en el proceso del censo, tu programa debe elegir la acción que desea realizar e imprimir una línea en consecuencia:

Acción	Significado
`r` P	**Leer** la ubicación P. Después de imprimir esta línea, tu programa debe leer una línea con el valor actual escrito en P.
`w` P V	**Escribir** en la ubicación P el nuevo valor V. Si varias instancias escriben en la misma P el mismo día, obtendrás el veredicto *Not correct*. Excepto en los ejemplos y la subtarea 3, debes escribir $0 \leq V \leq 1$; consulta la sección de Puntuación.
`!` N	**Responder y parar**: informar que hay N integrantes y dejar de participar en el censo. Después de responder, **tu programa debe terminar normalmente**. (Ten en cuenta que otras instancias de tu programa pueden seguir ejecutándose durante días adicionales antes de responder y terminar.)

Si alguna instancia de tu programa responde el valor incorrecto de N, viola el protocolo, utiliza más de 500 días, o excede el límite de tiempo/memoria (por proceso), tu envío será juzgado como *Not correct* para el caso de prueba dado.

De lo contrario, tu programa será *(Partially) Correct* en el caso de prueba y será puntuado basándose en el valor D: el número máximo de días que cualquier instancia tardó en responder. Para obtener la puntuación completa, necesitas resolver cada caso de prueba con $D \leq 61$ y $V \leq 1$. Consulta la sección de Puntuación para más detalles.

Vaciado (Flushing). Si no estás utilizando las plantillas proporcionadas, asegúrate de vaciar la salida estándar (flush) después de imprimir cada línea, de lo contrario tu programa podría ser juzgado como *Not correct*. En Python, esto sucede automáticamente si usas `input()` para leer líneas. En C++, `cout << endl;` vacía además de imprimir un salto de línea; si usas printf, usa `fflush(stdout)`.

Restricciones

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- Puedes usar como máximo 500 días.

Puntuación

Tu programa será probado en varios casos de prueba agrupados en subtareas. Para obtener la puntuación de una subtarea, debes resolver correctamente todos los tests que contiene.

- **Subtask 0 [0 puntos]**: Ejemplos (puedes escribir cualquier entero $0 \leq V \leq 1\,000\,000\,000$).
- **Subtask 1 [11 puntos]**: $M \leq 100$, y las N integrantes tienen IDs $0, 1, ..., N-1$.
- **Subtask 2 [12 puntos]**: $1 \leq N \leq 2$.
- **Subtask 3 [22 puntos]**: $M \leq 8000$, y puedes escribir cualquier entero $0 \leq V \leq 1\,000\,000\,000$.
- **Subtask 4 [55 puntos]**: Sin restricciones adicionales.

En las subtareas 1, 2 y 4, solo puedes escribir $V = 0$ o $V = 1$ en cada acción de Escritura.

Sea X_s la puntuación máxima para la subtarea s (mostrada arriba), y D_s el mayor número de días que cualquiera de tus programas utiliza en un test en la subtarea s. Entonces:

$$\text{puntuación}_s = \begin{cases} X_s & \text{si } D_s \leq 61 \\ X_s \cdot \left(0.2 + 0.8 \cdot 1.01^{(60-D_s)}\right) & \text{si } 61 < D_s \leq 500 \\ 0 & \text{si } 500 < D_s. \end{cases}$$

El valor de puntuación_s se redondea al entero más cercano por subtarea, y tu puntuación total es la suma de estas. Para la puntuación completa, necesitas $D \leq 61$ y $V \leq 1$ en cada caso de prueba.

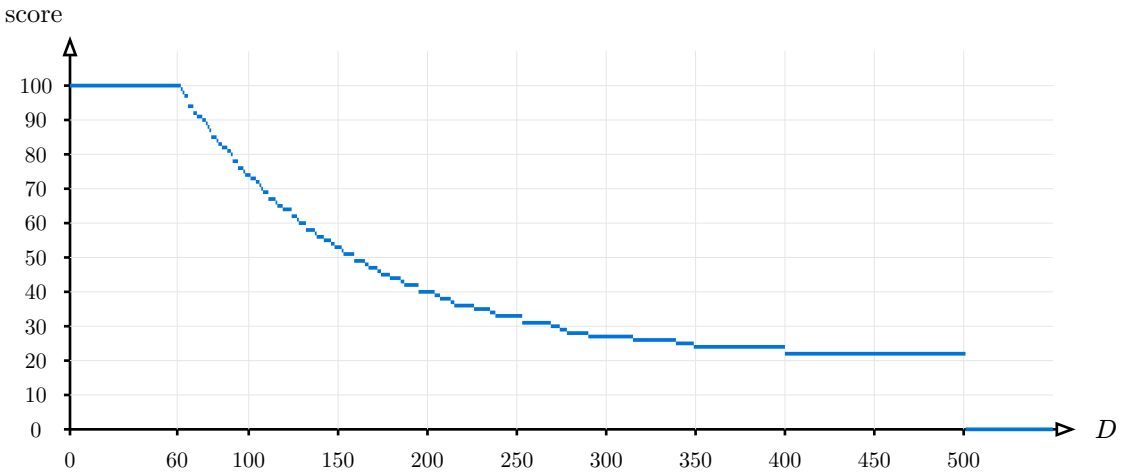

Figura 1: Puntuación total, asumiendo que cada subtarea se resuelve con el mismo D máximo.

Ejemplos de entrada/salida

Primer ejemplo. Cada par de columnas muestra la comunicación entre el evaluador y una instancia.

Gra.	Inst. 0	Gra.	Inst. 1	Gra.	Inst. 2	Gra.	Inst. 3	Gra.	Inst. 4
0 100		1 100		2 100		3 100		4 100	
	w 12 1		w 50 1		w 99 0		w 7 1		r 5
								0	
	r 50		r 7		r 12		w 1 1		! 5
1		1		1					
	! 5		r 1		w 0 0		! 5		
		1							
			! 5		! 5				

Segundo ejemplo.

Evaluador	Instancia 0
0 8000	
	w 0 0
	w 1 1
	r 2
1	
	! 2

Evaluador	Instancia 1
3 8000	
	w 2 1
	r 1
0	
	r 2
1	
	r 1
1	
	! 2

Explicación

Primer Ejemplo. Tenemos $N = 5$ integrantes con IDs consecutivos $0, 1, 2, 3, 4$ y $M = 100$ (válido para las subtareas 1, 3 y 4). La solución i corresponde a la integrante con ID i. La interacción anterior es solo una posible secuencia legal de operaciones y **no** pretende ser una estrategia eficiente o sensata; se muestra solo para ilustrar cómo funciona el protocolo.

Segundo Ejemplo. Tenemos $N = 2$ integrantes, con IDs 0 y 3, y $M = 8000$ (válido para las subtareas 2, 3 y 4). El primer día, la integrante con ID 0 escribe un 0 en la ubicación 0 (sin cambios), y la integrante con ID 3 escribe un 1 en la ubicación 2.

location	0	1	2	3	4	...
value	0	0	1	0	0	...

El segundo día, la ID 0 escribe un 1 en la ubicación 1, y la ID 3 lee esa misma ubicación. Ten en cuenta que la lectura ocurre durante el día, antes de la escritura por la tarde. Por lo tanto, la ID 3 todavía ve un 0.

location	0	1	2	3	4	...
value	0	1	1	0	0	...

El tercer día, ambas leen la ubicación 2, donde hay escrito un 1.

El cuarto día, la ID 0 responde que hay 2 integrantes (correcto), mientras que la ID 3 lee el 1 en la ubicación 1. La ID 0 sale inmediatamente después de esto y no participa en los días venideros.

Finalmente, en el día $D = 5$, la integrante restante también responde correctamente $N = 2$.

Testing

Para facilitar la prueba de tu solución, proporcionamos una herramienta sencilla que puedes descargar desde el CMS. El uso de la herramienta es opcional. Ten en cuenta que el evaluador oficial en el CMS es diferente de la herramienta de prueba.

Para usar la herramienta necesitas un archivo de entrada. Puedes usar los ejemplos de entrada proporcionados `census.input0.txt` y `census.input1.txt`, o crear los tuyos propios. El archivo de entrada debe comenzar con el número de integrantes N y los IDs posibles M, seguido de una línea con N números que especifiquen los IDs de las integrantes de la sociedad.

Para programas en Python, digamos `census.py` (normalmente ejecutado como `pypy3 census.py`) ejecuta la herramienta de prueba de la siguiente manera:

```
python3 testing_tool.py pypy3 census.py < census.input0.txt
```

Para programas en C++, primero compila tu solución:

```
g++ -DEVAL -std=gnu++20 -O2 -pipe -static -s -o census census.cpp
```

y luego ejecuta la herramienta de prueba:

```
python3 testing_tool.py ./census < census.input0.txt
```

Ten en cuenta que en este problema se utiliza la salida estándar para la comunicación con el evaluador, por lo que no debe usarse para depurar. En su lugar, puedes usar la salida de error estándar (stderr). En C++ puedes usar `cerr << msg << endl;`. En Python puedes usar `print(msg, file=sys.stderr)`.

La herramienta de prueba leerá y presentará estos mensajes de stderr junto con las consultas realizadas por todas las instancias de tu programa. Ten en cuenta que, por razones técnicas, pueden aparecer ligeramente desincronizados entre sí.