D. Censo (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 mujeres informáticas. Esta sociedad es muy secreta; ningún miembro conoce a las demás. Cada miembro tiene un ID único: un entero no negativo I.

La única comunicación entre las miembros es indirecta, mediante números escritos con tiza en diferentes ubicaciones por toda la ciudad. Cada 100 años, la sociedad realiza un censo para contar a sus miembros. Después de completar el censo, cada miembro debe conocer el número total de miembros en la sociedad.

El censo se lleva a cabo durante varios días. Cada día, cada miembro que aún participe en el proceso elegirá y realizará exactamente una acción: **read** (leer), **write** (escribir) o **stop** (dejar) de participar.

- Si un miembro elige **read**, selecciona una ubicación P. Durante el día, visita la ubicación P y lee el número que hay escrito allí.
- Si una miembro elige **write**, selecciona una ubicación P y un número V. Al final de la tarde, visita la ubicación P y cambia el número que había escrito allí por V. Como ya está oscuro, no puede leer el número anterior antes de escribir el nuevo.
- Si una miembro elige **stop**, ya no realiza ninguna acción en los días siguientes.

Si una miembro ve a otra escribir un número, podría reconocerla. Por lo tanto, está estrictamente prohibido que dos o más miembros elijan escribir en la misma ubicación el mismo día. (No existe tal restricción para leer, ya que eso se puede hacer discretamente.)

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

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

Implementación

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

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

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

Cuando se inicia una instancia de tu programa, primero debe leer una línea con dos enteros, I y M ($0 \le I \le M - 1$): el ID único de la miembro de la sociedad representada por esta instancia y el número total de IDs posibles. Dentro de cada caso de prueba, todas las instancias obtendrán el mismo

valor M y valores distintos de I. Ten en cuenta que puede haber IDs que no estén asignados a ninguna miembro.

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

Acción	Significado
`r` P	**read** (**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	**write** (**Escribir**) en la ubicación P el nuevo valor V. Si múltiples instancias escriben en el mismo P el mismo día, obtendrás el veredicto *No correcto*. Excepto por los ejemplos y la subtarea 3, debes escribir $0 \leq V \leq 1$; consulta la sección de Puntuación.
`!` N	**Responder y detenerse**: reportar que hay N miembros 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 que respondan y terminen.)

Si alguna instancia de tu programa responde el valor incorrecto de N, viola el protocolo, usa más de 500 días, o excede el límite de tiempo/memoria (por proceso), tu envío será juzgado como *No correcto* para el caso de prueba dado.

De lo contrario, tu programa será *(Parcialmente) Correcto* en el caso de prueba y será puntuado basándose en el valor D: el número máximo de días que cualquier instancia tomó para responder. Para 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.

Flushing. Si no estás usando las plantillas provistas, asegúrate de limpiar (flush) la salida estándar después de imprimir cada línea, o de lo contrario tu programa podría ser juzgado como *No correcto*. En Python, esto ocurre automáticamente si usas `input()` para leer líneas. En C++, `cout << endl;` limpia la salida 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 todas las pruebas 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 miembros 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 los puntos máximos para la subtarea s (mostrada arriba), y D_s el mayor número de días que cualquiera de tus programas usa en una prueba de la subtarea s. Entonces:

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

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

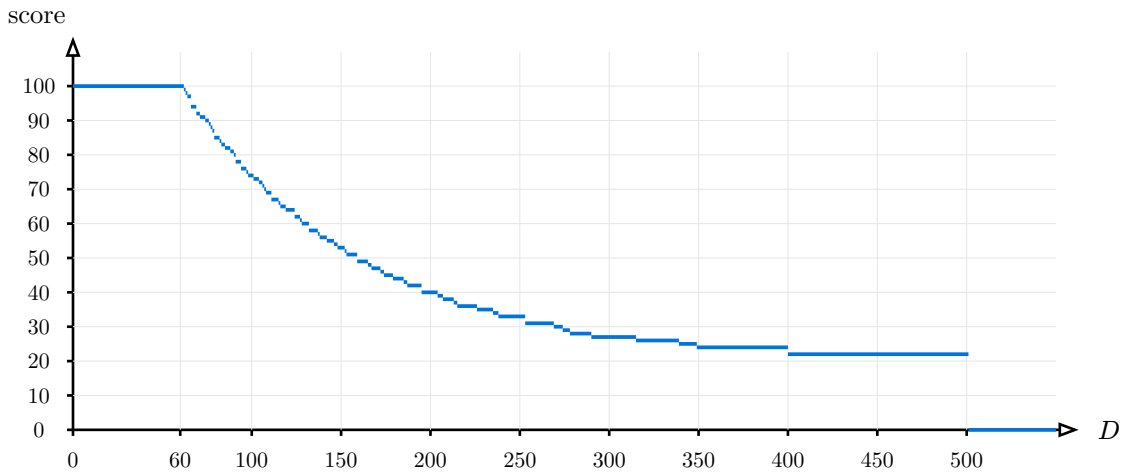

Figura 1: Puntuación total, asumiendo que cada subtarea se resuelve con el mismo D máximo.

Ejemplos de entrada/salida

Gra.	Sol. 0
0 100	
	w 12 1
	r 50
1	
	! 5

Gra.	Sol. 1
1 100	
	w 50 1
	r 7
1	
	r 1
1	
	! 5

Gra.	Sol. 2
2 100	
	w 99 0
	r 12
1	
	w 0 0
	! 5

Gra.	Sol. 3
3 100	
	w 7 1
	w 1 1
	! 5

Gra.	Sol. 4
4 100	
	r 5
0	
	! 5

Grader	Sol 0
0 8000	
	w 0 0
	w 1 1
	r 2
1	
	! 2

Grader	Sol 1
3 8000	
	w 2 1
	r 1
0	
	r 2
1	
	r 1
1	
	! 2

Explicación

Primer ejemplo. Tenemos $N = 5$ miembros con IDs consecutivos $0, 1, 2, 3, 4$ y $M = 100$ (válido para las subtareas 1, 3 y 4). La instancia i corresponde a la miembro con ID i. La interacción de arriba 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 $N = 2$ miembros, con IDs 0 y 3, y $M = 8000$ (válido para las subtareas 2, 3 y 4). En el primer día, la miembro con ID 0 escribe un 0 en la ubicación 0 (sin cambios), y la miembro con ID 3 escribe un 1 en la ubicación 2.

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

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

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

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

En el cuarto día, ID 0 responde que hay 2 miembros (correcto), mientras que ID 3 lee el 1 en la ubicación 1. ID 0 termina inmediatamente después de esto y no participa en los días siguientes.

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

Pruebas

Para facilitar la prueba de tu solución, proporcionamos una herramienta simple que puedes descargar desde CMS. La herramienta es de uso 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 provistos `census.input0.txt` y `census.input1.txt`, o crear los tuyos. El archivo de entrada debe comenzar con el número de miembros N y los IDs posibles M, seguido de una línea con N números que especifican los IDs de las miembros 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
```