

## D. Censo (census)

Limite de tempo: 1 segundos

Limite de memória: 128 MiB

Um fato pouco conhecido sobre Cesenatico é que ela é o lar de uma sociedade secreta de  $N$  informáticas. Esta sociedade é de fato muito secreta; nenhum membro conhece outro. Cada membro possui um ID único: um número inteiro não negativo  $I$ .

A única comunicação entre os membros é indireta, através de números rabiscados com giz em diferentes locais pela cidade. A cada 100 anos, a sociedade realiza um censo para contar seus membros. Após a conclusão do censo, cada membro deve saber o número total de membros na sociedade.

O censo ocorre ao longo de vários dias. A cada dia, todo membro que ainda está participando do processo escolherá e realizará exatamente uma ação: **ler**, **escrever** ou **parar** de participar.

- Se um membro escolhe **ler**, ela escolhe um local  $P$ . Durante o dia, ela visita o local  $P$  e lê o número que está escrito lá.
- Se um membro escolhe **escrever**, ela escolhe um local  $P$  e um número  $V$ . Tarde da noite, ela visita o local  $P$  e altera o número que estava escrito lá para  $V$ . Como já está escuro, ela não consegue ler o número antigo antes de escrever o novo.
- Se um membro escolhe **parar**, ela não realiza mais nenhuma ação nos dias seguintes.

Se um membro vir outro escrever um número, ela poderia reconhecê-la. Portanto, é estritamente proibido que dois ou mais membros escolham escrever no mesmo local no mesmo dia. (Não existe tal restrição para a leitura, pois ela pode ser feita discretamente.)

Se um ou mais membros lerem de um local onde outro membro deseja escrever no mesmo dia, todas as leituras ocorrem antes da escrita.

Como a sociedade deve planejar seu processo de censo para minimizar o número de dias até que todos saibam a contagem correta de membros?

### Implementação

⇒ Este é um problema interativo, no qual um número desconhecido de instâncias ( $1 \leq N \leq 100$ ) do seu programa será executado simultaneamente. Cada instância simula um membro da sociedade.

Existem  $10^{18}$  locais. O número  $P$  de um local deve satisfazer  $0 \leq P < 10^{18}$ . Inicialmente, o valor escrito em todos os locais é  $V = 0$ .

O novo valor  $V$  escrito em um local deve sempre ser um número inteiro tal que  $0 \leq V \leq 10^9$ . Na maioria das subtarefas,  $V$  só pode ser 0 ou 1. Veja a seção de Pontuação para mais detalhes.

Quando uma instância do seu programa inicia, ela deve primeiro ler uma linha com dois inteiros,  $I$  e  $M$  ( $0 \leq I \leq M - 1$ ): o ID único do membro da sociedade representado por esta instância e o número total de IDs possíveis. Dentro de cada caso de teste, todas as instâncias receberão o mesmo valor  $M$  e valores distintos de  $I$ . Observe que pode haver IDs que não são atribuídos a nenhum membro.

Então, para cada dia no processo de censo, seu programa deve escolher a ação que deseja realizar e imprimir uma linha de acordo:

Ação	Significado
$r\ P$	<b>Ler</b> local $P$ . Após imprimir esta linha, seu programa deve ler uma linha com o valor atual escrito em $P$ .
$w\ P\ V$	<b>Escrever</b> no local $P$ o novo valor $V$ . Se múltiplas instâncias escreverem no mesmo $P$ no mesmo dia, você receberá o veredito <i>Incorreto</i> . Exceto pelos exemplos e subtarefa 3, você deve escrever $0 \leq V \leq 1$ ; veja a seção de Pontuação.
$! N$	<b>Responder e parar:</b> informe que existem $N$ membros e pare de participar do censo. Após responder, <b>seu programa deve encerrar normalmente</b> . (Observe que outras instâncias do seu programa podem continuar rodando por dias adicionais antes de responderem e encerrarem.)

Se qualquer instância do seu programa responder o valor errado de  $N$ , violar o protocolo, usar mais de 500 dias, ou exceder o limite de tempo/memória (por processo), sua submissão será julgada como *Incorreto* para o caso de teste dado.

Caso contrário, seu programa será (*Parcialmente*) *Correto* no caso de teste e pontuado com base no valor  $D$ : o número máximo de dias que qualquer instância levou para responder. Para a pontuação máxima, você precisa resolver cada caso de teste com  $D \leq 61$  e  $V \leq 1$ . Veja a seção de Pontuação para detalhes.

**Flush.** Se você não estiver usando os modelos fornecidos, certifique-se de dar “flush” na saída padrão após imprimir cada linha, ou seu programa pode ser julgado como *Incorreto*. Em Python, isso acontece automaticamente se você usar `input()` para ler linhas. Em C++, `cout << endl;` faz o flush além de imprimir uma nova linha; se usar `printf`, use `fflush(stdout)`.

## Restrições

- $1 \leq N \leq 100$ .
- $1 \leq M \leq 100\,000$ .
- Você pode usar no máximo 500 dias.

## Pontuação

Seu programa será testado em vários casos de teste agrupados em subtarefas. Para obter a pontuação de uma subarefa, você deve resolver corretamente todos os testes nela contidos.

- **Subtarefa 0 [ 0 pontos]:** Exemplos (você pode escrever qualquer inteiro  $0 \leq V \leq 1\,000\,000\,000$ ).
- **Subtarefa 1 [11 pontos]:**  $M \leq 100$ , e os  $N$  membros possuem IDs  $0, 1, \dots, N - 1$ .
- **Subtarefa 2 [12 pontos]:**  $1 \leq N \leq 2$ .
- **Subtarefa 3 [22 pontos]:**  $M \leq 8000$ , e você pode escrever qualquer inteiro  $0 \leq V \leq 1\,000\,000\,000$ .
- **Subtarefa 4 [55 pontos]:** Sem restrições adicionais.

**Nas subtarefas 1, 2 e 4, você só pode escrever  $V = 0$  ou  $V = 1$  em cada ação de Escrita.**

Seja  $X_s$  a pontuação máxima para a subarefa  $s$  (mostrada acima), e  $D_s$  o maior número de dias que qualquer um dos seus programas usa em um teste na subarefa  $s$ . Então:

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

O valor da  $\text{pontuação}_s$  é arredondado para o inteiro mais próximo por subtarefa, e sua pontuação total é a soma destas. Para obter a pontuação total da tarefa, você precisa de  $D \leq 61$  e  $V \leq 1$  em todos os casos de teste.

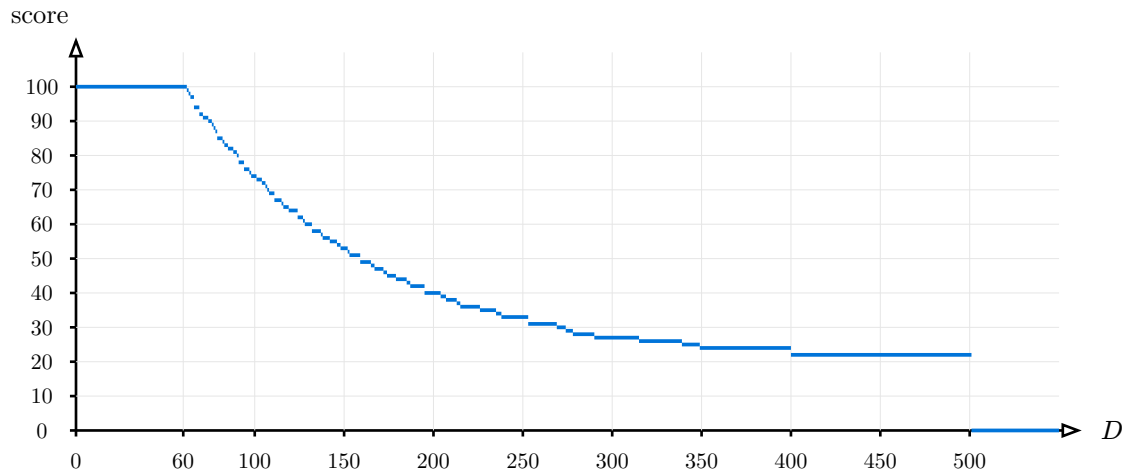


Figura 1: Pontuação total, assumindo que cada subtarefa seja resolvida com o mesmo  $D$  máximo.

## Exemplos

Primeiro exemplo. Cada par de colunas mostra a comunicação entre o grader e uma instância.

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 exemplo.

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

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

## Explicação

**Primeiro Exemplo.** Temos  $N = 5$  membros com IDs consecutivos 0, 1, 2, 3, 4 e  $M = 100$  (válido para as subtarefas 1, 3 e 4). A instância  $i$  corresponde ao membro com ID  $i$ . A interação acima é apenas uma possível sequência legal de operações e **não** pretende ser uma estratégia eficiente ou sensata; ela é mostrada apenas para ilustrar como o protocolo funciona.

**Segundo Exemplo.** Temos  $N = 2$  membros, com IDs 0 e 3, e  $M = 8000$  (válido para as subtarefas 2, 3 e 4). No primeiro dia, o membro com ID 0 escreve 0 no local 0 (sem alteração), e o membro com ID 3 escreve 1 no local 2.

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

No segundo dia, o ID 0 escreve 1 no local 1, e o ID 3 lê esse mesmo local. Observe que a leitura acontece durante o dia, antes da escrita à noite. Portanto, o ID 3 ainda vê um 0.

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

No terceiro dia, ambos leem o local 2, onde um 1 está escrito.

No quarto dia, o ID 0 responde que há 2 membros (correto), enquanto o ID 3 lê o 1 no local 1. O ID 0 sai imediatamente após isso e não participa nos dias seguintes.

Finalmente, no dia  $D = 5$ , o membro restante também responde corretamente  $N = 2$ .

## Testes

Para facilitar o teste da sua solução, fornecemos uma ferramenta simples que você pode baixar do CMS. O uso da ferramenta é opcional. Observe que o grader oficial no CMS é diferente da ferramenta de teste.

Para usar a ferramenta, você precisa de um arquivo de entrada. Você pode usar os exemplos de entrada fornecidos `census.input0.txt` e `census.input1.txt`, ou criar o seu próprio. O arquivo de entrada deve começar com o número de membros  $N$  e IDs possíveis  $M$ , seguido por uma linha com  $N$  números especificando os IDs dos membros da sociedade.

Para programas em Python, digamos `census.py` (normalmente executado como `pypy3 census.py`) execute a ferramenta de teste da seguinte forma:

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

Para programas em C++, primeiro compile sua solução:

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

e então execute a ferramenta de teste:

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

Observe que, neste problema, a saída padrão é usada para comunicação com o grader, portanto, não deve ser usada para depuração. Em vez disso, você pode usar a saída de erro padrão (stderr). Em C++, você pode usar `cerr << msg << endl;`. Em Python, você pode usar `print(msg, file=sys.stderr)`.

A ferramenta de teste lerá e apresentará essas mensagens de stderr junto com as consultas realizadas por todas as instâncias do seu programa. Observe que, por razões técnicas, elas podem aparecer ligeiramente dessincronizadas entre si.