D. إحصاء (census)

الحد الزمني: 1 ثواني
حد الذاكرة: 128MiB

هناك معلومة غير معروفة حول تشيزيناتيكو هي أنها موطن لتجمع سري يضم N إناث مبرمجات. هذا التجمع سري جدا، حيث لاتوجد أي أنثى تعرف هوية عضوة أخرى

الطريقة الوحيدة للتواصل بين الأعضاء تتم بشكل غير مباشر، بواسطة أرقام تكتب بالطباشير في مواقع مختلفة في المدينة. كل 100 سنة، يتم إجراء عملية إحصاء لحساب عدد الأعضاء لهذا التجمع. عندما تنتهي عملية الإحصاء، يجب أن تعرف كل عضوة العدد الإجمالي لأعضاء هذا التجمع

يستغرق الإحصاء عدة أيام. كل يوم، تقوم كل عضوة التي لاتزال في طور المشاركة في هذه العملية بتنفيذ مهمة واحدة فقط: **تقرأ**، **تكتب**، أو **تتوقف** عن المشاركة

- إذا اختارت العضوة أن **تقرأ**، ستختار موقع P. خلال فترة النهار، تقوم بزيارة الموقع P و تقرأ العدد المكتوب هناك
- إذا اختارت العضوة أن **تكتب** ، ستختار موقع P و عدد V. خلال وقت متأخر من المساء، تقوم بزيارة الموقع P و تغير العدد المكتوب هناك إلى V. و بما أن الوقت يكون مظلم، لا تستطيع قراءة العدد القديم قبل أن تكتب العدد الجديج
- إذا اختارت العضوة أن **تتوقف**، لا يمكنها أن تقوم بأي عملية في الأيام المقبلة

إذا رأت عضوة عضوة أخرى تكتب رقم، يمكنها أن تتعف على هويتها. لهذا، يمنع منعا باتا لعضوتين أو أكثر اختيار الكتابة في نفس الموقع في نفس اليوم. (لا يوجد مثل هذا القيد على القراءة، حيث يمكن القيام بذلك بسرية وتكتم.)

إذا قامت عضوة واحدة أو أكثر بالقراءة من موقع تريد عضوة أخرى الكتابة فيه في نفس اليوم، فإن جميع عمليات القراءة تحدث قبل عملية الكتابة.

كيف ينبغي للتجمع أن تخطط لعملية الإحصاء الخاصة بها لتقليل عدد الأيام حتى تعرف كل عضوة العدد الصحيح للأعضاء؟

التنفيذ

⇨ هذه مسألة تفاعلية (Interactive problem)، حيث سيتم تشغيل عدد غير معروف من النسخ $(1 \leq N \leq 100)$ من برنامجك في وقت واحد. كل نسخة تحاكي عضوة واحدة من ها التجمع.

يوجد 10^{18} مواقع. العدد P للمواقع يجب أن يحقق $0 \leq P < 10^{18}$. في البداية, القيمة التي تكون مكتوبة في جميع المواقع هي $V = 0$.

القيمة الجديدة V المكتوبة على أي موقع يجب أن تكون دائما عدد صحيح بحيث $0 \leq V \leq 10^9$. في أغلب المهمات الفردية, V يكون إما 0 أو 1. انظر قسم التقييم (Scoring) لمزيد من التفاصيل.

، عندما تبدأ نسخة من برنامجك، يجب أن تقرأ أولاً سطرًا يحتوي على عددين صحيحين I و M $(0 \leq I \leq M - 1)$:

المعرف الفريد (ID) للعضوة التي تمثلها هذه النسخة، وإجمالي عدد المعرفات الممكنة. داخل كل حالة اختبار (Test case)، ستحصل جميع النسخ على نفس القيمة M و قيم I مختلفة

لاحظ أنه قد تكون هناك معرفات (IDs) غير مخصصة لأي عضوة. بعد ذلك، ولكل يوم في عملية الإحصاء، يجب على برنامجك اختيار الإجراء الذي يريد تنفيذه وطباعة سطر وفقًا لذلك:

الإجراء	المعنى
Pr	قراءة الموقع P. بعد طباعة هذا السطر، يجب على برنامجك قراءة سطر يحتوي على القيمة الحالية المكتوبة في الموقع P.

V w P	**كتابة** القيمة الجديدة V في الموقع P. إذا قامت نسخ متعددة بالكتابة في نفس الموقع P في نفس اليوم، فستحصل على حكم غير صحيح. باستثناء الأمثلة والمهمة الفرعية 3، يجب كتابة $0 \leq V \leq 1$؛ انظر قسم التقييم.
N !	**الإجابة والتوقف:** الإبلاغ بأن هناك N من الأعضاء والتوقف عن المشاركة في الإحصاء. بعد الإجابة، **يجب أن يخرج برنامجك بشكل طبيعي**. (لاحظ أن النسخ الأخرى من برنامجك قد تستمر في العمل لأيام إضافية قبل أن تجيب وتخرج.)

إذا أجابت أي نسخة من برنامجك بقيمة خاطئة لـ N، أو انتهكت البروتوكول، أو استخدمت أكثر من 500 يوم، أو تجاوزت حد الوقت/الذاكرة (المخصص لكل عملية)، فسيتم الحكم على إرسالك بأنه غير صحيح لحالة الاختبار المعطاة.

بخلاف ذلك، سيكون برنامجك صحيحًا *(جزئيًا)* في حالة الاختبار ويتم تقييمه بناءً على القيمة D: الحد الأقصى لعدد الأيام التي استغرقتها أي نسخة للإجابة. للحصول على الدرجة الكاملة، تحتاج إلى حل كل حالة اختبار بحيث يكون $D \leq 61$ و $V \leq 1$. انظر قسم التقييم للحصول على التفاصيل.

تفريغ المخرجات (Flushing). إذا كنت لا تستخدم القوالب المقدمة، فتأكد من تفريغ المخرجات القياسية بعد طباعة كل سطر، وإلا فقد يتم الحكم على برنامجك بأنه غير صحيح. في لغة Python، يحدث هذا تلقائيًا إذا استخدمت `input()` لقراءة السطور. في لغة C++، تؤدي `cout << endl;` إلى التفريغ بالإضافة إلى طباعة سطر جديد؛ وإذا كنت تستخدم printf، فاستخدم `fflush(stdout)`.

القيود

- $1 \leq N \leq 100$.
- $1 \leq M \leq 100\,000$.
- يمكنك استخدام 500 يوم كحد أقصى.

توزيع الدرجات

سيتم اختبار برنامجك على عدة حالات اختبار مقسمة إلى مهام فرعية. للحصول على درجة مهمة فرعية، يجب عليك حل جميع الاختبارات التي تحتوي عليها بشكل صحيح.

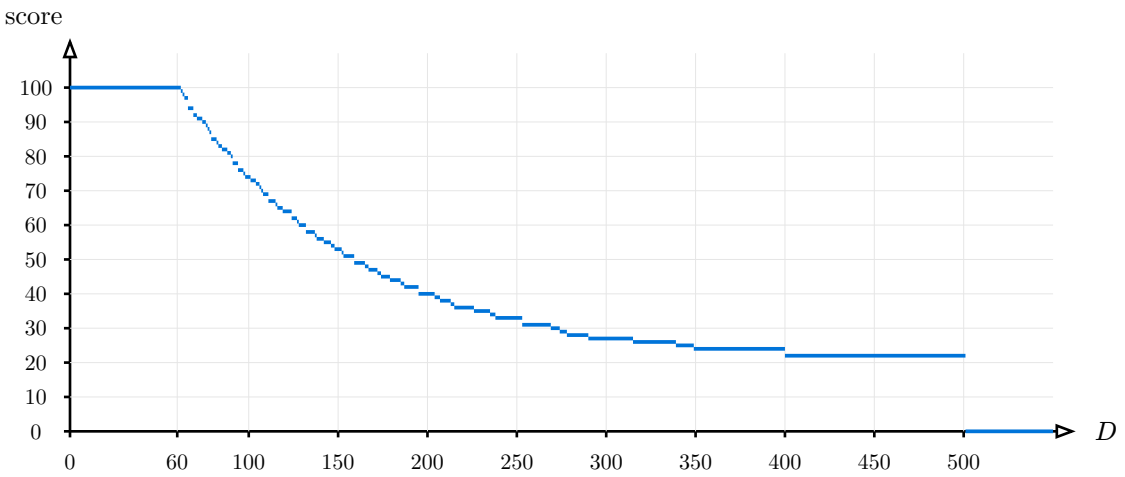
- **المهمة الفرعية 0 [0 نقاط]**: أمثلة (يمكنك كتابة أي عدد صحيح $0 \leq V \leq 1\,000\,000\,000$).
- **المهمة الفرعية 1 [11 نقاط]**: $M \leq 100$، والأعضاء الـ N لديهم المعرفات $0, 1, ..., N-1$.
- **المهمة الفرعية 2 [12 نقاط]**: $1 \leq N \leq 2$.
- **المهمة الفرعية 3 [22 نقاط]**: $M \leq 8000$، ويمكنك كتابة أي عدد صحيح $0 \leq V \leq 1\,000\,000\,000$.
- **المهمة الفرعية 4 [55 نقاط]**: لا توجد قيود إضافية.

في المهام الفرعية 1 و 2 و 4، يمكنك فقط كتابة $V = 0$ أو $V = 1$ في كل إجراء كتابة.

لتكن X_s هي الدرجة القصوى للمهمة الفرعية s (الموضحة أعلاه)، و D_s هي أكبر عدد من الأيام تستخدمه أي من برامجك في اختبار ضمن المهمة الفرعية s. إذن:

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

يتم تقريب قيمة score_s إلى أقرب عدد صحيح لكل مهمة فرعية، وتكون درجتك الإجمالية هي مجموع هذه الدرجات. للحصول على الدرجة الكاملة، تحتاج إلى $D \leq 61$ و $V \leq 1$ في كل حالة اختبار.



شكل 1: الدرجة الإجمالية، على افتراض أن كل مهمة فرعية قد حُلّت بنفس الحد الأقصى D.

أمثلة للإدخال/الإخراج

المثال الأول. يوضح كل زوج من الأعمدة التواصل بين المصحح ونسخة واحدة.

مصحح	نسخة 0	مصحح	نسخة 1	مصحح	نسخة 2	مصحح	نسخة 3	مصحح	نسخة 4
100 0		100 1		100 2		100 3		100 4	
	w 12 1		w 50 1		w 99 0		w 7 1		r 5
	r 50		r 7		r 12		w 1 1		5 !
1		1		1		5 !		0	
	5 !		r 1		w 0 0				
		1		5 !					
		5 !							

المثال الثاني.

المصحح	النسخة 0	المصحح	النسخة 1
8000 0		8000 3	
	w 0 0		w 2 1
	w 1 1		r 1
		0	
	r 2		r 2
1		1	
	2 !		r 1
		1	
			2 !

الشرح

المثال الأول. لدينا $N = 5$ أعضاء بمعرفات متتالية $0, 1, 2, 3, 4$ و $M = 100$ (صالح للمهام الفرعية 1 و 3 و 4). الحل i يتوافق مع العضو ذي المعرف i. التفاعل الموضح أعلاه هو مجرد تسلسل قانوني واحد ممكن للعمليات وليس المقصود منه أن يكون إستراتيجية فعالة أو منطقية؛ تم عرضه فقط لتوضيح كيفية عمل البروتوكول.

المثال الثاني. لدينا $N = 2$ من الأعضاء، بمعرفات 0 و 3، و $M = 8000$ (صالح للمهام الفرعية 2 و 3 و 4). في اليوم الأول، يكتب العضو ذو المعرف 0 القيمة 0 في الموقع 0 (لا تغيير)، ويكتب العضو ذو المعرف 3 القيمة 1 في الموقع 2.

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

في اليوم الثاني، يكتب المعرف 0 القيمة 1 في الموقع 1، ويقرأ المعرف 3 نفس هذا الموقع. لاحظ أن القراءة تحدث خلال النهار، قبل الكتابة في المساء. وبالتالي، لا يزال المعرف 3 يرى القيمة 0.

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

في اليوم الثالث، يقرأ كلاهما الموقع 2، حيث توجد القيمة 1 مكتوبة.

في اليوم الرابع، يجيب المعرف 0 بأن هناك عضوين 2 (وهو صحيح)، بينما يقرأ المعرف 3 القيمة 1 في الموقع 1. يخرج المعرف 0 فورًا بعد ذلك ولا يشارك في الأيام القادمة.

أخيرًا، في اليوم $D = 5$، يجيب العضو المتبقي أيضًا بشكل صحيح $N = 2$.

Testing

لتسهيل اختبار حلك، نقدم أداة اختبار بسيطة يمكنك تنزيلها من نظام CMS. استخدام الأداة اختياري. لاحظ أن المصحح الرسمي على CMS يختلف عن أداة الاختبار.

لاستخدام الأداة، تحتاج إلى ملف مدخلات. يمكنك استخدام ملفات المدخلات النموذجية المقدمة `census.input0.txt` و `census.input1.txt`، أو إنشاء ملفات خاصة بك. يجب أن يبدأ ملف المدخلات بعدد الأعضاء N والمعرفات الممكنة M، متبوعًا بسطر يحتوي على N من الأرقام تحدد معرفات أعضاء الجمعية.

بالنسبة لبرامج Python، لنقل `census.py` (والذي يتم تشغيله عادةً باستخدام `pypy3 census.py`) قم بتشغيل أداة الاختبار كما يلي:

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

بالنسبة لبرامج C++، قم أولاً بتصنيف حلك:

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

ثم قم بتشغيل أداة الاختبار:

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

لاحظ أنه في هذه المسألة يتم استخدام المخرجات القياسية للتواصل مع المصحح، لذا لا ينبغي استخدامها لتصحيح الأخطاء. بدلاً من ذلك، يمكنك استخدام مخرج الأخطاء القياسي (stderr). في لغة C++ يمكنك استخدام `cerr << msg << endl;`. وفي لغة Python يمكنك استخدام `print(msg, file=sys.stderr)`.

ستقوم أداة الاختبار بقراءة وعرض رسائل stderr هذه جنبًا إلى جنب مع الاستعلامات التي أجرتها جميع نسخ برنامجك. لاحظ أنه لأسباب تقنية، قد تظهر غير متزامنة تمامًا مع بعضها البعض.