سجلات الإزاحة

يعمل المهندس كريستوفر على نوع جديد من المعالجات الحاسوبية.

يستطيع المعالج الوصول إلى m وحدة ذاكرة مختلفة تتألف كل منها من b خانة ثنائية bit (حيث $m = 100$ و $b = 2000$)، والتي تسمى **سجلات** وهي مرقمة من 0 إلى $m - 1$.

نشير للسجلات المختلفة $r[0], r[1], \ldots, r[m-1]$. كل سجل هو مصفوفة من b خانة ثنائية مرقمة من 0 (الخانة اليمينية) إلى $b - 1$ (الخانة اليسارية). من اجل i $(0 \leq i \leq m - 1)$ و j $(0 \leq j \leq b - 1)$، نشير للخانة رقم j من السجل رقم i بـ $r[i][j]$.

من أجل أي تسلسل من الخانات الثنائية $d_0, d_1, \ldots, d_{l-1}$ (من اي طول l)، تكون **القيمة الصحيحة (integer)** للتسلسل هي $2^0 \cdot d_0 + 2^1 \cdot d_1 + \ldots + 2^{l-1} \cdot d_{l-1}$. فتكون **القيمة الصحيحة المخزنة في السجل** i هي القيمة الصحيحة لتسلسل الخانات الثنائية المخزنة فيه، أي $2^0 \cdot r[i][0] + 2^1 \cdot r[i][1] + \ldots + 2^{b-1} \cdot r[i][b-1]$.

هنالك 9 أنواع من **التعليمات** لهذا المعالج تستخدم لتعديل الخانات الثنائية في السجلات. تعمل كل تعليمة مع واحد أو أكثر من السجلات وتخزن نتيجتها في سجل واحد. في ما يلي، نستخدم التعبير $x := y$ لنشير إلى تغير قيمة x لتصبح مساوية للقيمة y. في ما يلي وصف للعمليات التي تتم في كل نوع من التعليمات.

- $move(t, y)$: نسخ مصفوفة الخانات الثنائية من السجل y إلى السجل t. من أجل كل j $(0 \leq j \leq b - 1)$، $r[t][j] := r[y][j]$.
- $store(t, v)$: تعديل قيمة السجل t لتصبح مساوية لـ v، حيث v مصفوفة من b خانة ثنائية. من أجل كل j $(0 \leq j \leq b - 1)$، $r[t][j] := v[j]$.
- $and(t, x, y)$: اجراء عملية bitwise-AND للسجلين x و y، وتخزين النتيجة في السجل t. من أجل j $(0 \leq j \leq b - 1)$، تكون $r[t][j] := 1$ إذا كان **كلا** $r[x][j]$ و $r[y][j]$ قيمة 1، وتكون $r[t][j] := 0$ فيما عدا ذلك.
- $or(t, x, y)$: اجراء عملية bitwise-OR للسجلين x و y، وتخزين النتيجة في السجل t. من أجل j $(0 \leq j \leq b - 1)$، تكون $r[t][j] := 1$ إذا كان **لأحد** $r[x][j]$ أو $r[y][j]$ على الأقل قيمة 1، وتكون $r[t][j] := 0$ فيما عدا ذلك.
- $xor(t, x, y)$: اجراء عملية bitwise-XOR للسجلين x و y، وتخزين النتيجة في السجل t. من أجل j $(0 \leq j \leq b - 1)$، تكون $r[t][j] := 1$ إذا كان **لأحد** $r[x][j]$ أو $r[y][j]$ فقط القيمة 1، وتكون $r[t][j] := 0$ فيما عدا ذلك.
- $not(t, x)$: اجراء عملية bitwise-NOT للسجل x، وتخزين النتيجة في السجل t. من أجل j $(0 \leq j \leq b - 1)$، $r[t][j] := 1 - r[x][j]$.
- $left(t, x, p)$: ازاحة جميع خانات السجل x إلى اليسار p ازاحة، وتخزين النتيجة في السجل t. تكون نتيجة عملية الازاحة في السجل x إلى اليسار p ازاحة هي المصفوفة v المكونة من b خانة ثنائية. من أجل j $(0 \leq j \leq b - 1)$، $v[j] = r[x][j - p]$ إذا كان $j \geq p$، و $v[j] = 0$ فيما عدا ذلك. من أجل j $(0 \leq j \leq b - 1)$، $r[t][j] := v[j]$.

- $right(t, x, p)$: ازاحة جميع خانات السجل x إلى اليمين p ازاحة، وتخزين النتيجة في السجل t . تكون نتيجة عملية الازاحة في السجل x إلى اليمين p ازاحة هي المصفوفة v المكونة من b خانة ثنائية. من أجل j $(0 \le j \le b-1)$ ، $v[j] = r[x][j+p]$ إذا كان $j \le b-1-p$ ، و $v[j] = 0$ فيما عدا ذلك. من أجل j $(0 \le j \le b-1)$ ، $r[t][j] := v[j]$.

- $add(t, x, y)$: جمع القيم الصحيحة المخزنة في السجل x والسجل y ، و تخزين النتيجة في السجل t . إن عملية الجمع تتم بباقي قسمة 2^b . بشكل رسمي، لتكن X هي القيمة الصحيحة المخزنة في السجل x ، و Y هي القيمة الصحيحة المخزنة في السجل y قبل تنفيذ التعليمة. لتكن T القيمة الصحيحة المخزنة في السجل t بعد تنفيذ التعليمة. في حالة $X + Y < 2^b$ ، تكون قيمة خانات t هي $T = X + Y$. فيما عدا ذلك، تكون قيمة خانات t هي $T = X + Y - 2^b$.

يرغب كرستوفر بحل نوعين من المهام باستخدام المعالج الجديد. يشار إلى نوع المهمة بالعدد الصحيح s . من أجل كلا نوعي المهام، عليك انتاج **برنامجاً** مؤلفاً من تسلسل من التعليمات المعرفة أعلاه.

يتألف **دخل** برنامجك من n عدداً صحيحاً $a[0], a[1], \ldots, a[n-1]$ ، لكل منها k خانة صحيحة، أي $a[i] < 2^k$ ($0 \le i \le n-1$). قبل بدء تنفيذ برنامجك، تخزن جميع ارقام الدخل بشكل تسلسلي في السجل 0 ، فمن أجل كل i $(0 \le i \le n-1)$ تكون القيمة الصحيحة لتسلسل الـk-خانة ثنائية $r[0][i \cdot k], r[0][i \cdot k + 1], \ldots, r[0][(i+1) \cdot k - 1]$ مساويةً للقيمة $a[i]$. لاحظ أن $n \cdot k \le b$. تكون جميع الخانات المتبقية في السجل 0 (أي الخانات ذات الدليل index بين$n \cdot k$ و $b - 1$ ، ضمناً) بالإضافة إلى جميع الخانات الثنائية في جميع السجلات الأخرى مساوية للـ0 .

يتألف تشغيل برنامجك من تنفيذ تعليماته بالترتيب. بعد تنفيذ آخر تعليمة، يتم حساب **الخرج** من برنامجك بالاعتماد على آخر قيمة لخانات السجل 0 . تفصيلاً، يتكون الخرج من سلسلة من n قيمة صحيحة $c[0], c[1], \ldots, c[n-1]$ ، حيث تكون من أجل كل i ($0 \le i \le n-1$)، تكون قيمة $c[i]$ هي القيمة الصحيحة لتسلسل من الخانات الثنائية $i \cdot k$ to $(i+1) \cdot k - 1$ من السجل 0 . لاحظ أنه بعد تشغيل برنامجك تكون الخانات الثنائية المتبقية من السجل 0 (والتي لها دلائل مساوية أو أكبر من $n \cdot k$) بالإضافة إلى جميع الخانات الثنائية في جميع السجلات الأخرى مساوية لقيم اعتباطية arbitrary.

- المهمة الأولى $(s = 0)$ هي البحث عن اصغر قيمة صحيحة ضمن مجموعة من قيم دخل صحيحة $a[0], a[1], \ldots, a[n-1]$. تفصيلاً، يجب أن تكون قيمة $c[0]$ هي القيمة الصغرى من بين $a[0], a[1], \ldots, a[n-1]$. يمكن للقيم $c[1], c[2], \ldots, c[n-1]$ أن تكون اعتباطية.

- المهمة الثانية $(s = 1)$ هي ترتيب قيم الدخل الصحيحة $a[0], a[1], \ldots, a[n-1]$ بترتيب غير متناقص. تفصيلاً، من أجل i $(0 \le i \le n-1)$، يجب أن تكون $c[i]$ مساوية للعدد الصحيح ذو الترتيب $1 + i$ بين الأرقام الصحيحة $a[0], a[1], \ldots, a[n-1]$ بعد ترتيبها. (أي أن $c[0]$ هو الرقم الصحيح الأصغر بين أرقام الدخل).

قم بتزويد كريستوفر ببرامج، مؤلف كل منها من q تعليمة على الاكثر، تقوم بحل هاتين المهمتين.

تفاصيل التنجيز

يجب عليك تنجيز الإجرائية التالية:

```
void construct_instructions(int s, int n, int k, int q)
```

- s : نوع المهمة.
- n : عدد قيم الدخل الصحيحة.
- k : عدد الخانات الثنائية لكل قيمة دخل صحيحة.
- q : العدد الأكبر من التعليمات المسموح بها.
- يتم استدعاء هذه الإجرائية لمرة واحدة فقط ويجب ان تؤلف سلسلة من التعليمات لإنجاز المهمة المطلوبة.

يجب على الإجرائية أن تستدعي واحداً أو أكثر من الاجرائيات التالية لتأليف سلسلة التعليمات:

```
void append_move(int t, int y)
void append_store(int t, bool[] v)
void append_and(int t, int x, int y)
void append_or(int t, int x, int y)
void append_xor(int t, int x, int y)
void append_not(int t, int x)
void append_left(int t, int x, int p)
void append_right(int t, int x, int p)
void append_add(int t, int x, int y)
```

- تقوم كل اجرائية بإضافة تعليمة: $move(t, y)$ $store(t, v)$, $and(t, x, y)$, $or(t, x, y)$, $xor(t, x, y)$, $not(t, x)$, $left(t, x, p)$, $right(t, x, p)$ و $add(t, x, y)$ إلى برنامجك، بالترتيب.
- من أجل كل التعليمات ذات الصلة، يجب أن تكون قيم t , x , y على الأقل 0 وعلى الأكثر $m - 1$.
- من أجل كل التعليمات ذات الصلة، t , x , y ليست مختلفة عن بعضها البعض بالضرورة.
- من أجل تعليمات $left$ و $right$ ،يجب أن تكون قيم p على الأقل 0 وعلى الأكثر b .
- من أجل تعليمات $store$ ، يكون طول v مساوياً لـ b .

يمكنك ايضاً استدعاء الإجراء التالي لمساعدتك على اختبار حلك

```
void append_print(int t)
```

- سيتم اهمال أي استدعاء لهذه الاجرائية خلال عملية تصحيح حلك.
- في المصحح النموذجي، تضيف هذه الإجرائية تعليمة $print(t)$ إلى برنامجك.
- عندما يصادف المصحح النموذجي تعليمة $print(t)$ خلال تنفيذ برنامجك، سيقوم بطباعة n عدداً صحيحاً كل منها مكون من k - خانة ثنائية، مشكّلة من أول $n \cdot k$ خانة ثنائية من السجل t (لمزيد من التفاصيل، انظر إلى المصحح النموذجي).
- يجب أن تحقق t الشرط $0 \leq t \leq m - 1$.
- لا يحتسب استدعاء هذه الاجرائية من عدد التعليمات الناتجة.

بعد إضافة آخر تعليمة، يجب أن تتوقف `construct_instructions`. بعدها يتم تقييم برنامجك على عدد من حالات الاختبار، تحدد كل حالة اختبار دخلاً مكوناً من n عدداً صحيحاً كل منها k -خانة ثنائية $a[0], a[1], \ldots, a[n-1]$. يعتبر حلك ناجحاً من أجل حالة الاختبار إذا كان خرج برنامجك $c[0], c[1], \ldots, c[n-1]$ من أجل الدخل المحدد محققاً للشروط التالية:

- إذا كان $s = 0$ ، فإن $c[0]$ يجب أن تكون الأصغر بين $a[0], a[1], \ldots, a[n-1]$.
- إذا كان $s = 1$ ، فمن أجل كل i $(0 \leq i \leq n-1)$ يجب أن تكون $c[i]$ هي القيمة ذات الترتيب $1 + i$ بين $a[0], a[1], \ldots, a[n-1]$ بعد ترتيبها.

يمكن للمصحح أن ينتج واحدة من رسائل الخطأ التالية أثناء تصحيح حلك:

- `Invalid index`: تم ارسال دليل سجل غير مقبول (ربما سالب) كبراميتر t أو x أو y لاستدعاء لاحدى الإجرائيات.
- `Value to store is not b bits long`: طول المصفوفة v الممرة لـ`append_store` لا تساوي b .
- `Invalid shift value`: قيمة p الممرة لـ `append_left` أو `append_right` ليست بين 0 و b ضمناً.
- `Too many instructions`: تحاول اجرائيتك إضافة اكثر من q تعليمة.

أمثلة

مثال 1

بفرض $s = 0$, $n = 2$, $k = 1$, $q = 1000$. هنالك قيمتين صحيحتين في الدخل $a[0]$ و $a[1]$ ، كل منها مكونة من $k = 1$ خانة ثنائية. قبل بدء تنفيذ البرنامج، $r[0][0] = a[0]$ و $r[0][1] = a[1]$. جميع بقية الخانات الثنائية في المعالج لها القيمة 0 . بعد تنفيذ جميع التعليمات في برنامجك، نحتاج لأن تكون $c[0] = r[0][0] = \min(a[0], a[1])$ ، والتي تعبر عن القيمة الأصغر بين $a[0]$ و $a[1]$.

هنالك 4 حالات دخل محتملة لبرنامجك:

- $a[0] = 0, a[1] = 0 : 1$
- $a[0] = 0, a[1] = 1 : 2$
- $a[0] = 1, a[1] = 0 : 3$
- $a[0] = 1, a[1] = 1 : 4$

نلاحظ أنه من أجل جميع الحالات الـ4، تكون $\min(a[0], a[1])$ مساوية لـbitwise-AND بين $a[0]$ و $a[1]$. وبالتالي، تكون إحدى الحلول الممكنة هي تأليف برنامج يقوم بالاستدعاءات التالية:

1. `append_move(1, 0)`, والتي تقوم بإضافة تعليمة تقوم بعملية نسخ من $r[0]$ إلى $r[1]$.
2. `append_right(1, 1, 1)`، والتي تقوم بإضافة تعليمة تزيح كل خانات السجل $r[1]$ خانة واحدة يميناً وتخزن النتيجة في نفس السجل $r[1]$. بما أن قيم الدخل مكونة من خانة ثنائية واحدة، تكون النتيجة في $r[1][0]$ مساوية لـ $a[1]$.
3. `append_and(0, 0, 1)`، والتي تقوم باضافة تعليمة bitwise-AND بين $r[0]$ و $r[1]$ ، وتخزن النتيجة في السجل $r[0]$. بعد تنفيذ هذه التعليمة، تصبح قيمة $r[0][0]$ هي bitwise-AND بين $r[0][0]$ و $r[1][0]$ ، والمساوية للـbitwise-AND بين $a[0]$ و $a[1]$ ، وهو المطلوب.

مثال 2

بفرض $s = 1$, $n = 2$, $k = 1$, $q = 1000$. وكما هو في المثال السابق، هنالك 4 حالات دخل محتملة لبرنامجك فقط. من أجل الحالات الأربعة، تكون $\min(a[0], a[1])$ مساوية لـbitwise-AND بين $a[0]$ و $a[1]$. و تكون $\max(a[0], a[1])$ مساوية لـbitwise-OR بين $a[0]$ و $a[1]$. وبالتالي، تكون إحدى الحلول الممكنة هي تأليف برنامج يقوم بالاستدعاءات التالية:

1. `append_move(1,0)`
2. `append_right(1,1,1)`
3. `append_and(2,0,1)`
4. `append_or(3,0,1)`
5. `append_left(3,3,1)`
6. `append_or(0,2,3)`

وبعد تنفيذ هذه التعليمات، يحتوي $c[0] = r[0][0]$ على القيمة $\min(a[0], a[1])$ ويحتوي $c[1] = r[0][1]$ على القيمة $\max(a[0], a[1])$ ، مما يرتب الدخل.

القيود

- $m = 100$
- $b = 2000$
- $0 \leq s \leq 1$
- $2 \leq n \leq 100$
- $1 \leq k \leq 10$

- $q \leq 4000$
- $0 \leq a[i] \leq 2^k - 1$ (من أجل $0 \leq i \leq n - 1$)

المسائل الجزئية

1. $s = 0, n = 2, k \leq 2, q = 1000$ (10 علامات)
2. $s = 0, n = 2, k \leq 2, q = 20$ (11 علامة)
3. $s = 0, q = 4000$ (12 علامة)
4. $s = 0, q = 150$ (25 علامة)
5. $s = 1, n \leq 10, q = 4000$ (13 علامة)
6. $s = 1, q = 4000$ (29 علامة)

المصحح النموذجي

يقرأ المصحح النموذجي الدخل على الشكل التالي:

- السطر 1 : $s\ n\ k\ q$

يليه عدد من الأسطر، يوصّف كل منها حالة اختبار. لكل حالة اختبار الشكل التالي:

- $a[0]\ a[1]\ \ldots\ a[n-1]$

وتوصف حالة اختبار لها الدخل المكون من n عدداً صحيحاً $a[0], a[1], ..., a[n-1]$. يلي توصيف كل حالات الاختبار سطراً يحتوي القيمة -1 فقط.

يقوم المصحح النموذجي باستدعاء `construct_instructions(s, n, k, q)` أولاً. إذا لم يحقق هذا الاستدعاء أي من القيود الموصوفة في نص المسألة، يطبع المصحح إحدى رسائل الخطأ المذكورة في نهاية قسم "تفاصيل التنجيز" ويتوقف. وفيما عدا ذلك، يطبع المصحح أولاً كل تعليمة تمت إضافتها ضمن `construct_instructions(s, n, k, q)` بالترتيب. من أجل تعليمات $store$ يتم طباعة v من الدليل 0 إلى الدليل $b - 1$. ومن ثم، يعالج المصحح النموذجي حالات الاختبار بالترتيب. ومن أجل كل حالة اختبار، يقوم بتشغيل برنامجك على قيم الدخل لحالة الاختبار.

من أجل تعليمة $print(t)$، ولتكن $d[0], d[1], \ldots, d[n-1]$ سلسلة من الأعداد الصحيحة، حيث من اجل كل i ($0 \leq i \leq n - 1$) تكون $d[i]$ هي القيمة الصحيحة لسلسة من الخانات الثنائية من $i \cdot k$ إلى $(i + 1) \cdot k - 1$ من السجل t (عند تنفيذ التعليمة). يطبع المصحح هذه السلسلة بالشكل التالي: `register` t : $d[0]\ d[1]\ \ldots\ d[n-1]$.

بعد تنفيذ جميع التعليمات، يطبع المصحح النموذجي خرج برنامجك.

في حالة $s = 0$ ، يكون خرج المصحح النموذجي من أجل كل حالة اختبار:

- $c[0]$.

في حالة $s = 1$ ، يكون خرج المصحح النموذجي من أجل كل حالة اختبار:

- $c[0]\ c[1]\ \ldots\ c[n-1]$.

بعد تنفيذ جميع حالات الاختبار، يطبع المصحح `number of instructions`: X حيث X هي عدد التعليمات في برنامجك.