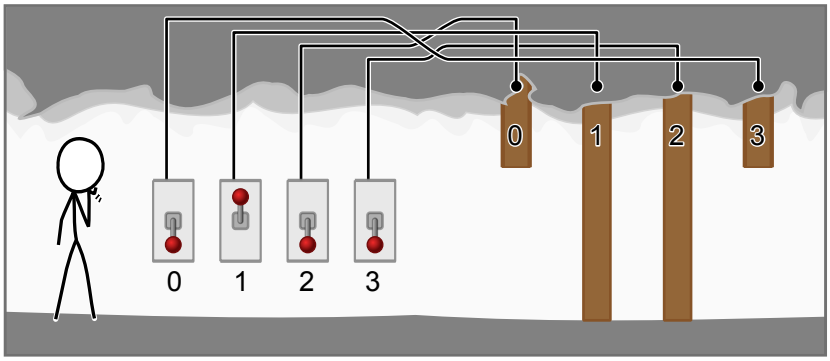
International Olympiad in Informatics 2013

July 2013 6-13
Brisbane, Australia
Day 2 tasks

غار

Persian — ۱.۰

بعد از گم کردن راه در یک پیاده‌روی طولانی از مرکز دانشگاه UQ، شما به ورودی یک مجموعه غار سحرآمیز رسیده‌اید که به اعماق زیر دانشگاه راه دارد. ورودی غار با یک سامانه‌ی امنیتی متشکل از N درب متوالی و N سوئیچ مسدود شده است. هر درب پشت درب قبلی قرار گرفته و هر سوئیچ به یک درب جداگانه وصل شده است (هیچ دو سوئیچی به یک درب متصل نیستند).



درب‌ها به ترتیب با اعداد 0, 1, ..., (N - 1) شماره‌گذاری شده‌اند، طوری که درب شماره‌ی 0 نزدیک‌ترین درب به شما می‌باشد. سوئیچ‌ها نیز با اعداد 0, 1, ..., (N - 1) شماره‌گذاری شده‌اند، اما شما نمی‌دانید که کدام سوئیچ به کدام درب متصل است.

سوئیچ‌ها در ورودی غار قرار گرفته‌اند. هر سوئیچ می‌تواند در یکی از دو حالت بالا یا پایین قرار داشته باشند. برای هر سوئیچ، تنها یکی از این حالات درست است: اگر یک سوئیچ در حالت درست خود باشد، دربی که به سوئیچ وصل است باز می‌شود و اگر سوئیچ در حالت درست خود نباشد، دربی که به آن وصل است، بسته خواهد بود. حالت درست می‌تواند برای سوئیچ‌های مختلف، متفاوت باشد و شما از حالت درست برای سوئیچ‌ها اطلاع ندارید.

شما می‌خواهید این سیستم امنیتی را بشناسید. برای این منظور، شما می‌توانید سوئیچ‌ها را در یک وضعیت دل‌خواه قرار دهید و سپس وارد غار شوید تا ببینید اولین درب بسته کدام است. درب‌ها شفاف نیستند و وقتی شما به اولین درب بسته برسید، هیچ دربی که پشت آن باشد را نمی‌بینید.

شما به اندازه‌ای زمان دارید که می‌توانید حداکثر 70,000 ترکیب از سوئیچ‌ها را بررسی کنید (و نه بیشتر). وظیفه شما این است که وضعیت درست هر سوئیچ و همچنین دربی که به آن سوئیچ متصل است را بیابید.

پیاده‌سازی

شما باید یک فایل شامل پیاده‌سازی تابع `exploreCave()` را به سامانه‌ی داوری ارسال کنید. این تابع می‌تواند حداکثر `70000` بار تابع `tryCombination()` را فراخوانی کند، و باید با فراخوانی تابع `answer()` در سامانه داوری به اتمام برسد. این توابع در ادامه توضیح داده شده‌اند.

تابع مصحح: `tryCombination()`

C/C++
```
int tryCombination(int S[]);
```

Pascal
```
function tryCombination(var S: array of LongInt) : LongInt;
```

توضیحات

این تابع در برنامه‌ی مصحح پیاده‌سازی شده است. این تابع به شما اجازه می‌دهد که یک ترکیب از سوئیچ‌ها را امتحان کنید، و سپس با ورود به غار اولین درب بسته را پیدا کنید. اگر همه‌ی درب‌ها باز باشند، تابع مقدار `-1` را باز می‌گرداند. این تابع در زمان `O(N)` اجرا می‌شود، یعنی زمان اجرا در بدترین حالت متناسب با `N` خواهد بود.

این تابع باید حداکثر `70000` بار صدا زده شود.

پارامترها

- `S`: یک آرایه به طول `N` که وضعیت هر سوئیچ را مشخص می‌کند. `S[i]` مربوط به وضعیت سوئیچ `i`اُم است. مقدار `0` به معنی بالا بودن سوئیچ و مقدار `1` به معنی پایین بودن آن است.
- خروجی: شماره اولین دربی که بسته است. در صورت باز بودن همه درب‌ها مقدار `-1` برگردانده خواهد شد.

تابع مصحح: `answer()`

C/C++
```
void answer(int S[], int D[]);
```

Pascal
```
procedure answer(var S, D: array of LongInt);
```

توضیحات

شما باید هنگامی این تابع را فراخوانی کنید که ترکیبی از سوئیچ‌ها که همه‌ی درب‌ها را باز می‌کند و همچنین شماره‌ی دربی که به هر سوئیچ متصل است را یافته باشید.

پارامترها

- `S`: یک آرایه به طول `N` که وضعیت درست هر سوئیچ را مشخص می‌کند. فرمت آن مشابه آن‌چه در تابع `tryCombination()` که در بالا توضیح داده شد می‌باشد.
- `D`: یک آرایه به طول `N` که درب متصل به هر سوئیچ را مشخص می‌کند. به طور مشخص، عنصر `D[i]` باید شامل شماره‌ی دربی باشد که سوئیچ `i` ام به آن متصل است.
- خروجی: این تابع چیزی باز نمی‌گرداند، اما باعث خاتمه‌ی برنامه می‌شود.

تابع شما: `exploreCave()`

C/C++
```
void exploreCave(int N);
```

Pascal
```
procedure exploreCave(N: longint);
```

توضیحات

برنامه‌ی ارسالی شما باید این تابع را پیاده‌سازی کند.

این تابع باید از تابع `tryCombination()` در برنامه‌ی مصحح، برای پیدا کردن وضعیت درست برای هر سوئیچ و درب متصل به آن، استفاده کند. بعد از تعیین این موارد، تابع `answer()` باید فقط یک بار فراخوانی شود.

پارامترها

- `N`: تعداد سوئیچ‌ها و درب‌های درون غار.

اجرای نمونه

فرض کنید درب‌ها و سوئیچ‌ها در وضعیتی که در شکل بالا نشان داده شده، قرار داشته باشند.

فراخوانی تابع	خروجی	توضیح
`tryCombination([1, 0, 1, 1])`	`1`	به شکل توجه کنید. سوئیچ‌های ۰، ۲ و ۳ پایین هستند، اما سوئیچ ۱ بالا است. تابع عدد ۱ را برمی‌گرداند، به این معنی که درب شماره‌ی ۱ اولین درب بسته از سمت چپ است.
`tryCombination([0, 1, 1, 0])`	`3`	درب‌های ۰، ۱ و ۲ باز هستند، درحالی که درب ۳ بسته است.
`tryCombination([1, 1, 1, 0])`	`-1`	پایین بردن سوئیچ ۰ باعث می‌شود که همه درب‌ها باز شوند و در نتیجه تابع عدد ۱- را برمی‌گرداند.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	(برنامه پایان می‌یابد)	حدس می‌زنیم که ترکیب درست سوئیچ‌ها `[1, 1, 1, 0]` است و سوئیچ‌های ۰، ۱، ۲ و ۳ به ترتیب به درب‌های ۳، ۱، ۰ و ۲ متصل هستند.

محدودیت‌ها

- محدودیت زمان: ۲ ثانیه
- محدودیت حافظه: ۳۲ مگابایت
- $1 \le N \le 5,000$

زیرمسئله‌ها

زیرمسئله‌ها	امتیاز	محدودیت‌های اضافی ورودی
۱	۱۲	برای هر i، سوییچ i به درب i متصل است. وظیفه‌ی شما تنها پیدا کردن ترکیب درست برای سوئیچ‌ها است.
۲	۱۳	ترکیب درست همواره [0, 0, 0, …, 0] خواهد بود. شما تنها باید مشخص کنید که هر سوئیچ به کدام درب وصل است.
۳	۲۱	$N \le 100$
۴	۳۰	$N \le 2,000$
۵	۲۴	(بدون محدودیت اضافی)

آزمایش

مصححی که روی کامپیوتر شما قرار دارد، ورودی را از فایل cave.in می‌خواند، که این فایل باید به شکل زیر باشد:

- خط ۱: N
- خط ۲: S[0] S[1] … S[N - 1]
- خط ۳: D[0] D[1] … D[N - 1]

اینجا N تعداد درب‌ها و سوئیچ‌ها، S[i] وضعیت درست برای سوئیچ i و D[i] دربی است که سوئیچ i به آن متصل است.

برای نمونه، مثال بالا باید به شکل زیر داده شود:

```
4
1 1 1 0
3 1 0 2
```

نکات زبان

C/C++ شما باید `#include "cave.h"` را به برنامه‌ی خود اضافه کنید.

Pascal شما باید `unit Cave` را تعریف کنید و همچنین توابع برنامه‌ی مصحح را به شکل زیر به برنامه‌ی خود اضافه کنید: `uses GraderHelpLib`. تمامی آرایه‌ها با شروع از `0` (و نه `1`) شمارش می‌شوند.

برای دیدن مثال‌ها به راه حل‌های نمونه بر روی کامپیوتر خود مراجعه کنید.

International Olympiad in Informatics 2013

July 2013 6-13

Brisbane, Australia

Day 2 tasks

روبـــات‌هـــا

Persian — ۱.۰

برادر کوچک ماریتا اسباب‌بازی‌هایش را کف اتاق رها کرده است! خوشبختانه ماریتا روبات‌های ویژه‌ای برای جمع‌آوری اسباب‌بازی‌ها طراحی کرده است. او به کمک شما برای تعیین آن که کدام روبات کدام اسباب‌بازی را جمع کند نیاز دارد.

تعداد `T` اسباب‌بازی داریم. اسباب‌بازی `i`اُم دارای وزن `W[i]` و اندازه‌ی `S[i]` است که هر دو اعداد صحیح هستند. روبات‌ها به یکی از دو دسته‌ی زیر تعلق دارند: (۱) دسته‌ی ضعیف (۲) دسته‌ی کوچک.

- تعداد `A` روبات ضعیف داریم. هر روبات ضعیف دارای محدودیت وزنی `X[i]` می‌باشد، به این معنی که فقط اسباب‌بازی با وزن کم‌تر از `X[i]` را می‌تواند حمل کند. برای روبات‌های ضعیف اندازه‌ی اسباب‌بازی‌ها اهمیتی ندارد.
- تعداد `B` روبات کوچک داریم. هر روبات کوچک دارای محدودیت اندازه‌ای `Y[i]` است، به این معنی که فقط اسباب‌بازی‌های با اندازه‌ی کم‌تر از `Y[i]` را می‌تواند حمل کند. وزن اسباب‌بازی‌ها برای روبات‌های کوچک اهمیتی ندارد.

هر روبات ماریتا به یک دقیقه زمان برای جمع‌آوری هر اسباب‌بازی نیاز دارد. روبات‌های متفاوت می‌توانند اسباب‌بازی‌های متفاوت را به طور هم‌زمان جمع‌آوری کنند.

شما باید تعیین کنید که آیا روبات‌های ماریتا می‌توانند همه‌ی اسباب‌بازی‌ها را جمع کنند، و اگر این گونه است کم‌ترین زمان لازم را محاسبه کنید.

مثال‌ها

به عنوان مثال اول، فرض کنید که تعداد `A = 3` روبات ضعیف با محدودیت وزنی `X = [6,2,9]`، و تعداد `B = 2` روبات کوچک با محدودیت اندازه‌ای `Y = [4, 7]` داریم. در ضمن فرض کنید تعداد اسباب‌بازی‌ها `T = 10` است.

شماره‌ی اسباب‌بازی	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
وزن	۴	۸	۲	۷	۱	۵	۳	۸	۷	۱۰
اندازه	۶	۵	۳	۹	۸	۱	۳	۷	۶	۵

کم‌ترین زمانی که می‌توان همه‌ی اسباب‌بازی‌ها را جمع کرد ۳ دقیقه است.

	روبات ضعیف ۰	روبات ضعیف ۱	روبات ضعیف ۲	روبات کوچک ۰	روبات کوچک ۱
دقیقه‌ی اول	اسباب‌بازی ۰	اسباب‌بازی ۴	اسباب‌بازی ۱	اسباب‌بازی ۶	اسباب‌بازی ۲
دقیقه‌ی دوم	اسباب‌بازی ۵		اسباب‌بازی ۳		اسباب‌بازی ۸
دقیقه‌ی سوم			اسباب‌بازی ۷		اسباب‌بازی ۹

به عنوان مثال دوم، فرض کنید تعداد `A = 2` روبات ضفیف با محدودیت وزنی `X = [2, 5]`، و تعداد `B = 1` روبات کوچک با محدودیت اندازه‌ای `Y = [2]` داریم. در ضمن فرض کنید تعداد اسباب‌بازی‌ها `T = 3` است.

شماره‌ی اسباب‌بازی	۰	۱	۲
وزن	۳	۵	۲
اندازه	۱	۳	۲

هیچ روباتی نمی‌تواند اسباب‌بازی با وزن ۵ و اندازه‌ی ۳ را بردارد. بنابراین جمع‌آوری همه‌ی اسباب‌بازی‌ها غیرممکن است.

پیاده‌سازی

شما باید تابع `putaway()` را در یک فایل به صورت زیر پیاده‌سازی و ارسال کنید:

تابع شما: `putaway()`

C/C++
```
int putaway(int A, int B, int T,
        int X[], int Y[], int W[], int S[]);
```

Pascal
```
function putaway(A, B, T : LongInt;
        var X, Y, W, S : array of LongInt) : LongInt;
```

توضیحات

این تابع باید کم‌ترین زمان ممکن (بر حسب دقیقه) را که روبات‌ها نیاز دارند تا تمام اسباب‌بازی‌ها را جمع‌آوری کنند محاسبه کند، یا عدد `-1` را به معنی غیر ممکن بودن برگرداند.

پارامترها

- `A`: تعداد روبات‌های ضعیف.
- `B`: تعداد روبات‌های کوچک.
- `T`: تعداد اسباب‌بازی‌ها.
- `X`: یک آرایه با اندازه‌ی `A` از اعداد صحیح که محدودیت وزنی هر روبات ضعیف را مشخص می‌کند.
- `Y` یک آرایه با اندازه‌ی `B` از اعداد صحیح که محدودیت اندازه‌ای هر روبات کوچک را مشخص می‌کند.

- `W`: یک آرایه با اندازه‌ی `T` از اعداد صحیح که وزن اسباب‌بازی‌ها را مشخص می‌کند.
- `S`: یک آرایه با اندازه‌ی `T` از اعداد صحیح که اندازه‌ی اسباب‌بازی‌ها را مشخص می‌کند.
- خروجی: کم‌ترین زمان موردنیاز (برحسب دقیقه) برای جمع کردن اسباب‌بازی‌ها، یا `-1` اگر چنین کاری امکان‌پذیر نباشد.

اجرای نمونه

اجرای زیر مربوط به مثال اول بالا است:

Parameter	Value
A	`3`
B	`2`
T	`10`
X	`[6, 2, 9]`
Y	`[4, 7]`
W	`[4, 8, 2, 7, 1, 5, 3, 8, 7, 10]`
S	`[6, 5, 3, 9, 8, 1, 3, 7, 6, 5]`
Returns	`3`

اجرای زیر مربوط به مثال دوم بالا است:

Parameter	Value
A	`2`
B	`1`
T	`3`
X	`[2, 5]`
Y	`[2]`
W	`[3, 5, 2]`
S	`[1, 3, 2]`
Returns	`-1`

محدودیت‌ها

- محدودیت زمان: ۳ ثانیه

- محدودیت حافظه: ۶۴ مگابایت
- `1 ≤ T ≤ 1,000,000`
- `0 ≤ A, B ≤ 50,000` و `1 ≤ A + B`
- `1 ≤ X[i], Y[i], W[i], S[i] ≤ 2,000,000,000`

زیرمسئله‌ها

زیرمسئله	امتیاز	محدودیت‌های ورودی اضافی
۱	۱۴	`T = 2` و `A + B = 2` (دقیقاً دو اسباب‌بازی و دو روبات)
۲	۱۴	`B = 0` (تمام روبات‌ها ضعیف‌اند)
۳	۲۵	`T ≤ 50` و `A + B ≤ 50`
۴	۳۷	`T ≤ 10,000` و `A + B ≤ 1,000`
۵	۱۰	(بدون محدودیت اضافی)

آزمایش

مصححی که روی کامپیوتر شما قرار دارد ورودی را از فایل `robots.in` می‌خواند، که این فایل باید به شکل زیر باشد:

- خط ۱: `A B T`
- خط ۲: `X[0] … X[A-1]`
- خط ۳: `Y[0] … Y[B-1]`
- `T` خط بعد: `W[i] S[i]`

برای نمونه، مثال اول بالا باید به شکل زیر داده شود:

```
3 2 10
6 2 9
4 7
4 6
8 5
2 3
7 9
1 8
5 1
3 3
8 7
7 6
10 5
```

اگر `A = 0` یا `B = 0` آن‌گاه خط متناظر (خط ۲ یا ۳) باید خالی باشد.

نکات زبان

C/C++ عبارت `#include "robots.h"` را باید به برنامه‌ی خود اضافه کنید.

Pascal باید `unit Robots` را تعریف کنید. تمام آرایه‌ها از `0` (و نه `1`) شروع می‌شوند.

برای دیدن مثال‌ها به راه‌حل‌های نمونه بر روی کامپیوتر خود مراجعه کنید.

International Olympiad in Informatics 2013
July 2013 6-13
Brisbane, Australia
Day 2 tasks

بـازی

Persian — ۱.۰

بازا و شازا دارند بازی می‌کنند. صفحه‌ی بازی جدولی از خانه‌ها با R سطر و C ستون است که سطرهای آن با اعداد 0 تا $R - 1$، و ستون‌های آن با اعداد 0 تا $C - 1$ شماره‌گذاری شده‌اند. خانه‌ای که در سطر P و ستون Q قرار گرفته را با `(P, Q)` نشان می‌دهیم. در هر خانه یک عدد صحیح غیرمنفی نوشته شده، و در شروع بازی تمام این اعداد صفر هستند.

بازی به صورت زیر انجام می‌شود. در هر زمان، بازا می‌تواند یکی از دو عمل زیر را انجام دهد:

- عددی که در خانه‌ی `(P, Q)` نوشته شده را تغییر دهد؛
- از شازا بخواهد که بزرگ‌ترین مقسوم‌علیه مشترک (ب.م.م) تمام اعدادی که درون یک ناحیه‌ی مستطیل شکل از خانه‌ها با دو گوشه‌ی `(P, Q)` و `(U, V)` قرار می‌گیرند را محاسبه کند (مستطیل شامل خانه‌های مشخص‌شده در گوشه‌ها نیز هست).

بازا قبل از آن که خسته شود و برای بازی کریکت از خانه بیرون رود، در مجموع $N_U + N_Q$ عمل (N_U تغییر خانه و N_Q پرسیدن ب.م.م) انجام می‌دهد.

وظیفه‌ی شما این است که جواب درست سؤال‌های پرسیده شده را برای شازا پیدا کنید.

مثال‌ها

فرض کنید $R = 2$ و $C = 3$، و بازا با تغییرات زیر شروع می‌کند:

- خانه‌ی `(0, 0)` را به ۲۰ تغییر می‌دهد؛
- خانه‌ی `(0, 2)` را به ۱۵ تغییر می‌دهد؛
- خانه‌ی `(1, 1)` را به ۱۲ تغییر می‌دهد.

20	0	15
0	12	0

جدول حاصل در شکل بالا نشان داده شده است. در ادامه، بازا ب.م.م مستطیل‌های زیر را می‌پرسد:

- مستطیلی با دو گوشه‌ی مقابل `(0, 0)` و `(0, 2)` : سه عدد موجود در این مستطیل عبارت‌اند از ۲۰، ۰ و ۱۵، که ب.م.م آن‌ها ۵ است.
- مستطیلی با دو گوشه‌ی مقابل `(0, 0)` و `(1, 1)` : چهار عدد موجود در این مستطیل عبارت‌اند از ۲۰، ۰، ۰ و ۱۲، که ب.م.م آن‌ها ۴ است.

اکنون فرض کنید که بازا تغییرات زیر را اعمال می‌کند:

- خانه‌ی `(0,1)` را به ۶ تغییر می‌دهد؛
- خانه‌ی `(1,1)` را به ۱۴ تغییر می‌دهد.

20	6	15
0	14	0

مستطیل جدید در شکل بالا نشان داده شده است. بازا در ادامه مجدداً ب.م.م مستطیل‌های زیر را می‌پرسد:

- مستطیلی با دو گوشه‌ی مقابل `(0, 0)` و `(0, 2)` : این بار سه عدد موجود در این مستطیل عبارت‌اند از ۲۰، ۶ و ۱۵، که ب.م.م آن‌ها ۱ است.
- مستطیلی با دو گوشه‌ی مقابل `(0, 0)` و `(1, 1)` : این بار چهار عدد موجود در این مستطیل عبارت‌اند از ۲۰، ۶، ۰ و ۱۴، که ب.م.م آن‌ها ۲ است.

در این مثال، بازا در مجموع $N_U = 5$ تغییر و $N_Q = 4$ پرسش انجام داده است.

پیاده‌سازی

شما باید یک فایل حاوی توابع `update()` ، `init()` و `calculate()` همان گونه که در زیر توضیح داده شده‌اند به سامانه‌ی داوری ارسال کنید.

برای کمک به شما، راه‌حل‌های نمونه بر روی کامپیوتر شما قرار داده شده‌اند (`game.cpp` ، `game.c` و `game.pas`). این فایل‌ها شامل یک تابع `gcd2(X, Y)` است که بزرگ‌ترین مقسوم‌علیه مشترک دو عدد صحیح نامنفی داده‌شده‌ی `X` و `Y` را محاسبه می‌کند. اگر `X = Y = 0` آن‌گاه `gcd2(X, Y)` نیز مقدار `0` را برمی‌گرداند.

این تابع به اندازه‌ی کافی سریع است که بتوان با آن نمره‌ی کامل را به دست آورد. مشخصا، زمان اجرای این تابع در بدترین حالت متناسب با `log(X + Y)` است.

تابع شما: `init()`

C/C++
```
void init(int R, int C);
```

Pascal
```
procedure init(R, C : LongInt);
```

توضیحات

برنامه‌ی ارسالی شما باید این تابع را پیاده‌سازی کند.

این تابع اندازه‌ی اولیه‌ی جدول را به شما می‌دهد و به شما اجازه می‌دهد تمام متغیرهای سراسری (global) و داده‌ساختارهای موردنیاز را مقداردهی اولیه کنید. این تابع تنها یک بار، و پیش از هر گونه فراخوانی `update()` یا `calculate()` فراخوانی می‌شود.

پارامترها

- `R` : تعداد سطرها.
- `C` : تعداد ستون‌ها.

تابع شما: `update()`

C/C++
```
void update(int P, int Q, long long K);
```

Pascal
```
procedure update(P, Q : LongInt; K : Int64);
```

توضیحات

برنامه‌ی ارسالی شما باید این تابع را پیاده‌سازی کند.

این تابع وقتی که بازا مقدار یک خانه‌ی جدول را تغییر می‌دهد فراخوانی می‌شود.

پارامترها

- `P` : شماره‌ی سطر خانه‌ی جدول ($0 \le P \le R - 1$).
- `Q` : شماره‌ی ستون خانه‌ی جدول ($0 \le Q \le C - 1$).
- `K` : عدد صحیح جدید در این خانه‌ی جدول ($0 \le K \le 10^{18}$). می‌تواند برابر با مقدار فعلی خانه باشد.

تابع شما: `calculate()`

C/C++
```
long long calculate(int P, int Q, int U, int V);
```

Pascal
```
function calculate(P, Q, U, V : LongInt) : Int64;
```

توضیحات

برنامه‌ی ارسالی شما باید این تابع را پیاده‌سازی کند.

این تابع باید بزرگ‌ترین مقسوم‌علیه مشترک تمام اعداد موجود در مستطیلی که با دو گوشه‌ی مقابل `(P, Q)` و `(U, V)` مشخص می‌شوند را محاسبه کند. این مستطیل شامل دو خانه‌ی `(P, Q)` و `(U, V)` نیز می‌شود.

اگر تمام اعداد موجود در مستطیل صفر بودند، تابع باید مقدار صفر برگرداند.

پارامترها

- `P`: شماره‌ی سطر بالاترین-چپ‌ترین خانه‌ی مستطیل (`0 ≤ P ≤ R - 1`).
- `Q`: شماره‌ی ستون بالاترین-چپ‌ترین خانه‌ی مستطیل (`0 ≤ Q ≤ C - 1`).
- `U`: شماره‌ی سطر پایین‌ترین-راست‌ترین خانه‌ی مستطیل (`P ≤ U ≤ R - 1`).
- `V`: شماره‌ی ستون پایین‌ترین-راست‌ترین خانه‌ی مستطیل (`Q ≤ V ≤ C - 1`).
- خروجی: ب.م.م تمام اعداد داخل مستطیل، یا `0` اگر تمام آن اعداد صفر باشند.

اجرای نمونه

اجرای زیر مربوط به مثال بالا است:

Function Call	Returns
`init(2, 3)`	
`update(0, 0, 20)`	
`update(0, 2, 15)`	
`update(1, 1, 12)`	
`calculate(0, 0, 0, 2)`	`5`
`calculate(0, 0, 1, 1)`	`4`
`update(0, 1, 6)`	
`update(1, 1, 14)`	
`calculate(0, 0, 0, 2)`	`1`
`calculate(0, 0, 1, 1)`	`2`

محدودیت‌ها

- محدودیت زمان: زیرمسئله‌ها را ببینید.
- محدودیت حافظه: زیرمسئله‌ها را ببینید.
- $1 \le R, C \le 10^9$
- $0 \le K \le 10^{18}$ ، که در آن `K` هر عدد صحیحی است که بازا در خانه‌ی جدول قرار می‌دهد.

زیرمسئله‌ها

برای دیدن زیرمسئله‌ها و پارامترهای آن‌ها به نسخه‌ی انگلیسی (بخش Subtasks) مراجعه کنید.

آزمایش

مصححی که روی کامپیوتر شما قرار دارد ورودی را از فایل `game.in` می‌خواند. این فایل باید به شکل زیر باشد:

- خط ۱: `R C N`
- `N` خط بعد: یک عمل در هر خط، به ترتیبی که عمل‌ها رخ می‌دهند.

خط مربوط به هر عمل باید به یکی از دو شکل زیر باشد:

- برای مشخص کردن `update(P, Q, K)` : `1 P Q K`

- برای مشخص کردن `calculate(P, Q, U, V)` : `2 P Q U V`

برای نمونه، مثال بالا باید به شکل زیر داده شود:

```
2 3 9
1 0 0 20
1 0 2 15
1 1 1 12
2 0 0 0 2
2 0 0 1 1
1 0 1 6
1 1 1 14
2 0 0 0 2
2 0 0 1 1
```

نکات زبان

C/C++ عبارت `#include "game.h"` را باید به برنامه‌ی خود اضافه کنید.

Pascal باید `unit Game` را تعریف کنید. تمام آرایه‌ها از `0` (و نه `1`) شروع می‌شوند.

از آن جایی که اعداد صحیح درون خانه‌ی جدول می‌توانند خیلی بزرگ باشند، توصیه می‌کنیم که کاربران `C/C++` از نوع `long long` ، و کاربران پاسکال از نوع `Int64` استفاده کنند.