پیام

عایشه و بَسما دو دوست هستند که با یکدیگر مکاتبه می‌کنند. عایشه پیامی به نام M دارد که یک دنباله‌ای از S بیت (یعنی صفر یا یک) است که او می‌خواهد به بسما بفرستد. عایشه با ارسال **بسته‌ها** به بسما ارتباط برقرار می‌کند. هر بسته یک دنباله‌ای از 31 بیت است که از 0 تا 30 شماره‌گذاری شده است. عایشه می‌خواهد پیام M را با ارسال تعدادی بسته به بسما ارسال کند.

متأسفانه، کلئوپاترا ارتباط بین عایشه و بسما را مختل کرده و می‌تواند بسته‌ها را **آلوده** کند. به این معنا که کلئوپاترا می‌تواند در هر بسته دقیقا 15 بیت را تغییر دهد. به طور خاص، آرایه‌ای به نام C وجود دارد که طول آن 31 است و هر عنصر آن می‌تواند 0 یا 1 باشد و معنای زیر را دارد:

- $C[i] = 1$ نشان می‌دهد که بیت در اندیس i اُم توسط کلئوپاترا قابل تغییر است. این اندیس‌ها را **کنترل شده** توسط کلئوپاترا می‌نامیم.
- $C[i] = 0$ نشان می‌دهد که بیت در اندیس i اُم توسط کلئوپاترا قابل تغییر نیست.

آرایه C دقیقا حاوی 15 عدد یک و 16 عدد صفر است. در حین ارسال پیام M، مجموعه شاخص‌های کنترل شده توسط کلئوپاترا برای تمام بسته‌ها ثابت می‌ماند. عایشه می‌داند دقیقاً کدام 15 اندیس توسط کلئوپاترا کنترل می‌شوند. بسما فقط می‌داند که 15 اندیس توسط کلئوپاترا کنترل می‌شوند اما نمی‌داند کدام اندیس‌ها هستند.

فرض کنید A یک بسته است که عایشه تصمیم به ارسال آن می‌گیرد (که آن را **بسته اصلی** می‌نامیم). فرض کنید B بسته‌ای است که بسما دریافت می‌کند (که آن را **بسته آلوده** می‌نامیم). برای هر i که $0 \le i < 31$:

- اگر کلئوپاترا بیت با اندیس i را کنترل نکند ($C[i] = 0$)، بسما بیت i را به همان صورتی که توسط عایشه ارسال شده دریافت می‌کند ($B[i] = A[i]$).
- در غیر این صورت، اگر کلئوپاترا بیت با شاخص i را کنترل کند ($C[i] = 1$)، مقدار $B[i]$ توسط کلئوپاترا تعیین می‌شود.

عایشه بلافاصله پس از ارسال هر بسته می‌فهمد که بسته آلوده متناظر چه بوده است.

پس از اینکه عایشه تمام بسته‌ها را ارسال کرد، بسما تمام بسته‌های آلوده را **به همان ترتیب که ارسال شده بودند** دریافت می‌کند و باید پیام اصلی M را بازسازی کند.

وظیفه شما این است که استراتژی‌ای ابداع و پیاده‌سازی کنید که به عایشه امکان دهد پیام M را به بسما ارسال کند، به طوری که بسما بتواند M را از بسته‌های آلوده بازیابی کند. به طور خاص، شما باید دو تابع پیاده‌سازی کنید. تابع اول اقدامات عایشه را انجام می‌دهد. این تابع پیامی به نام M و آرایه‌ای به نام C دریافت می‌کند و باید برخی بسته‌ها را برای انتقال پیام به بسما ارسال کند. تابع دوم اقدامات بسما را انجام می‌دهد. این تابع بسته‌های آلوده را دریافت کرده و باید پیام اصلی M را بازیابی کند.

جزئیات پیاده‌سازی

اولین تابعی که باید پیاده‌سازی کنید عبارت است از:

```
void send_message(std::vector<bool> M, std::vector<bool> C)
```

- M: یک آرایه به طول S که پیام عایشه برای ارسال به بسما را توصیف می‌کند.
- C: یک آرایه به طول 31 که شاخص‌های بیت‌های کنترل شده توسط کلئوپاترا را نشان می‌دهد.
- این تابع در هر تست **حداکثر 2100 بار** فراخوانی می‌شود.

این تابع باید تابع زیر را برای ارسال یک بسته فراخوانی کند:

```
std::vector<bool> send_packet(std::vector<bool> A)
```

- A: یک بسته اصلی (آرایه‌ای به طول 31) که بیت‌های ارسال شده توسط عایشه را نشان می‌دهد.
- این تابع یک بسته آلوده B را که نشان دهنده بیت‌های دریافتی توسط بسما است بازمی‌گرداند.
- این تابع می‌تواند حداکثر 100 بار در هر فراخوانی send_message فراخوانی شود.

دومین تابعی که باید پیاده‌سازی کنید عبارت است از:

```
std::vector<bool> receive_message(std::vector<std::vector<bool>> R)
```

- R: آرایه‌ای که بسته‌های آلوده را توصیف می‌کند. بسته‌ها از بسته‌های ارسال شده توسط عایشه در یک فراخوانی send_message سرچشمه می‌گیرند و **به همان ترتیبی که توسط عایشه ارسال شده‌اند** داده می‌شوند. هر عنصر از R یک آرایه به طول 31 است که نشان‌دهنده یک بسته آلوده است.
- این تابع باید یک آرایه S بیتی بازگرداند که با پیام اصلی M برابر است.
- در هر تست، این تابع ممکن است **چندین بار** فراخوانی شود، ولی **دقیقا یک بار** به ازای هر فراخوانیsend_message. **ترتیب** فراخوانی‌های تابع receive_message **لزومی ندارد که با ترتیب فراخوانی‌های مربوطه send_message یکسان باشد**.

توجه داشته باشید که در سیستم نمره‌دهی، تابع‌های send_message و receive_message در **دو برنامه جداگانه** فراخوانی می‌شوند.

محدودیت‌ها

- $1 \leq S \leq 1024$
- C دقیقا دارای 31 عنصر است که 16 تا از آنها برابر 0 و 15 تا از آنها برابر 1 هستند.

زیرمسئله‌ها و نمره‌دهی

اگر در هر تست، فراخوانی تابع send_packet با قوانین ذکر شده در بالا مطابقت نداشته باشد یا مقدار بازگشتی از هر یک از فراخوانی‌های تابع receive_message نادرست باشد، امتیاز راه‌حل شما برای آن تست 0 خواهد بود.

در غیر این صورت، فرض کنید Q حداکثر تعداد فراخوانی‌های تابع send_packet در بین تمام فراخوانی‌های send_message در تمامی تست‌ها باشد. همچنین فرض کنید X برابر باشد با:

- 1، اگر $Q \leq 66$ باشد.
- 0.95^{Q-66}، اگر $66 < Q \leq 100$ باشد.

سپس، نمره به صورت زیر محاسبه می‌شود:

زیرمسئله	امتیاز	محدودیت‌های اضافی
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	بدون محدودیت اضافی.

توجه داشته باشید که در برخی موارد رفتار تصحیح کننده ممکن است **انطباق‌پذیر** باشد. این به این معناست که مقادیر خروجی send_packet ممکن است نه تنها به آرگومان‌های ورودی آن بستگی داشته باشد بلکه به بسیاری چیزهای دیگر، از جمله ورودی‌ها و مقادیر خروجی از فراخوانی‌های قبلی این تابع و اعداد شبه تصادفی تولید شده توسط ارزیاب نیز بستگی داشته باشد. ارزیاب **قطعی** است به این معنا که اگر دو بار آن را اجرا کنید و در هر دو اجرا همان بسته‌ها را ارسال کنید، همان تغییرات را در آنها ایجاد خواهد کرد.

مثال

در نظر بگیرید فراخوانی زیر انجام شده است.

```
send_message([0, 1, 1, 0],
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

پیامی که عایشه تلاش می‌کند به بسما ارسال کند، $[0, 1, 1, 0]$ است. بیت‌هایی با اندیس‌های 0 تا 15 توسط کلئوپاترا قابل تغییر نیستند، در حالی که بیت‌های با اندیس‌های 16 تا 30 می‌توانند توسط کلئوپاترا تغییر کنند.

به عنوان مثال، فرض کنید رفتار کلئوپاترا قطعی است، و او بیت‌هایی که کنترل می‌کند را به صورت پی‌درپی با 0 و 1 به صورت متناوب پر می‌کند، یعنی او به اندیس اولی که کنترل می‌کند (شاخص 16 در این مثال) 0 اختصاص می‌دهد، به دومین اندیسی که کنترل می‌کند (شاخص 17) 1 می‌دهد، به سومین اندیسی که کنترل می‌کند (شاخص 18) 0 می‌دهد، و به همین ترتیب ادامه می‌دهد.

عایشه می‌تواند تصمیم بگیرد که دو بیت از پیام اصلی را در یک بسته به صورت زیر ارسال کند: او بیت اول را در اولین 8 اندیسی که کنترل می‌کند قرار می‌دهد و بیت دوم را در 8 اندیس بعدی‌ای که کنترل می‌کند قرار می‌دهد.

سپس عایشه تصمیم می‌گیرد بسته زیر را ارسال کند:

```
send_packet([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

توجه داشته باشید که کلئوپاترا می‌تواند بیت‌های 15 اندیس آخر را تغییر دهد، بنابراین عایشه می‌تواند آن‌ها را به دلخواه تنظیم کند، زیرا ممکن است بازنویسی شوند. با توجه به استراتژی فرض شده از کلئوپاترا، رویه مقدار زیر را بازمی‌گرداند: $[0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

عایشه تصمیم می‌گیرد که دو بیت آخر M را در بسته دوم به صورت مشابه قبلی ارسال کند:

```
send_packet([1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

با توجه به استراتژی فرض شده از کلئوپاترا، رویه مقدار زیر را بازمی‌گرداند: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

عایشه می‌تواند بسته‌های بیشتری ارسال کند، اما تصمیم می‌گیرد که ارسال نکند.

سپس ارزیاب فراخوانی زیر را انجام می‌دهد:

```
receive_message([[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                  0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
                 [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                  0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]])
```

بسما پیام M را به صورت زیر بازیابی می‌کند. از هر بسته، او اولین بیتی را که دو بار پشت سر هم تکرار شده است برمی‌دارد، و آخرین بیتی که دو بار پشت سر هم تکرار شده است. یعنی از بسته اول، او بیت‌های $[0,1]$ را برمی‌دارد و از بسته دوم بیت‌های $[1,0]$ را برمی‌دارد. با کنار هم قرار دادن آنها، او پیام $[0,1,1,0]$ را بازسازی می‌کند، که مقدار بازگشتی صحیح برای این فراخوانی `receive_message` است.

می‌توان نشان داد که با استراتژی فرض شده کلئوپاترا و برای پیام‌هایی با طول 4، این رویکرد بسما به درستی M را بازسازی می‌کند، بدون توجه به مقدار C. با این حال، این رویکرد در حالت کلی درست نیست.

ارزیاب نمونه

ارزیاب نمونه انطباق‌پذیر نیست. در عوض، رفتار کلئوپاترا قطعی است، و او بیت‌هایی که کنترل می‌کند را به صورت پی‌درپی با بیت‌های 0 و 1 به صورت متناوب پر می‌کند، همانطور که در مثال بالا توضیح داده شد.

فرمت ورودی: **خط اول ورودی شامل یک عدد صحیح T است که تعداد سناریوها را مشخص می‌کند.** T سناریو به دنبال آن می‌آیند. هر کدام از آنها در فرمت زیر ارائه می‌شوند:

```
S
M[0]  M[1]  ...  M[S-1]
C[0]  C[1]  ...  C[30]
```

فرمت خروجی: ارزیاب نمونه نتیجه هر یک از T سناریوها را به همان ترتیبی که در ورودی ارائه شده‌اند در فرمت زیر می‌نویسد:

```
K L
D[0]  D[1]  ...  D[L-1]
```

اینجا، K تعداد فراخوانی‌های send_packet است، D پیامی است که توسط receive_message بازگردانده شده است و L طول آن است.