Message

Aisha và Basma là hai người bạn có liên hệ với nhau. Aisha có một thông điệp M muốn gửi cho Basma, đó là một chuỗi S bit (tức là gồm các số 0 hoặc số 1). Aisha giao tiếp với Basma bằng cách gửi cho cô ấy **các gói tin**. Một gói tin là một chuỗi 31 bit được đánh chỉ số từ 0 đến 30. Aisha muốn gửi thông điệp M cho Basma bằng cách gửi cho cô ấy một số lượng gói tin nhất định.

Không may, Cleopatra đã can thiệp vào thông tin liên lạc giữa Aisha và Basma và có thể **sửa** các gói tin. Nghĩa là, trong mỗi gói tin Cleopatra có thể sửa các bit tại đúng 15 vị trí. Cụ thể, có một mảng C có độ dài 31, trong đó mỗi phần tử đều là 0 hoặc 1, với ý nghĩa sau:

- $C[i] = 1$ chỉ ra rằng bit có chỉ số i có thể được thay đổi bởi Cleopatra. Ta gọi những chỉ số này là **được kiểm soát** bởi Cleopatra.
- $C[i] = 0$ chỉ ra rằng bit có chỉ số i không thể được thay đổi bởi Cleopatra.

Mảng C chứa chính xác 15 số 1 và 16 số 0. Trong khi gửi thông điệp M, tập hợp các chỉ số do Cleopatra kiểm soát vẫn giữ nguyên cho tất cả các gói tin. Aisha biết chính xác 15 chỉ số nào do Cleopatra kiểm soát. Basma chỉ biết rằng có 15 chỉ số do Cleopatra kiểm soát, nhưng cô ấy không biết là những chỉ số nào.

Giả sử A là một gói tin mà Aisha quyết định gửi (ta gọi là **gói tin gốc**). Giả sử B là gói tin mà Basma nhận được (ta gọi là **gói tin bị sửa**). Đối với mỗi i, mà $0 \leq i < 31$:

- nếu Cleopatra không kiểm soát bit có chỉ số i ($C[i] = 0$), Basma nhận được bit i do Aisha gửi ($B[i] = A[i]$),
- trái lại, nếu Cleopatra kiểm soát bit có chỉ số i ($C[i] = 1$), giá trị của $B[i]$ do Cleopatra quyết định.

Ngay sau khi gửi mỗi gói tin, Aisha sẽ biết được gói tin bị sửa tương ứng là gì.

Sau khi Aisha gửi tất cả các gói tin, Basma nhận được tất cả các gói tin bị sửa **theo thứ tự gửi** và phải tái tạo lại thông điệp gốc M.

Nhiệm vụ của bạn là thiết kế và cài đặt một chiến lược cho phép Aisha gửi thông điệp M đến Basma, để Basma có thể khôi phục M từ các gói tin đã bị sửa. Cụ thể, bạn cần cài đặt hai hàm. Hàm đầu tiên thực hiện các hành động của Aisha. Hàm được truyền vào một thông điệp M và mảng C, và phải gửi một số gói tin để chuyển thông điệp đến Basma. Hàm thứ hai thực hiện các hành động của Basma. Hàm được truyền vào các gói tin đã bị sửa và phải khôi phục thông điệp gốc M.

Chi tiết cài đặt

Hàm đầu tiên bạn cần cài đặt là:

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

- M: mảng có độ dài S mô tả thông điệp mà Aisha muốn gửi đến Basma.
- C: mảng có độ dài 31 mô tả các chỉ số của các bit được Cleopatra kiểm soát.
- Hàm này có thể được gọi **tối đa 2100 lần** trong mỗi trường hợp test.

Hàm này sẽ gọi đến hàm sau để gửi một gói tin:

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

- A: gói tin gốc (mảng có độ dài 31) biểu diễn các bit được gửi bởi Aisha.
- Hàm này trả về một gói tin bị sửa B biểu diễn các bit sẽ được Basma nhận.
- Hàm này có thể được gọi tối đa 100 lần trong mỗi lệnh gọi `send_message`.

Hàm thứ hai bạn cần cài đặt là:

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

- R: mảng mô tả các gói tin bị sửa. Các gói tin bắt nguồn từ các gói tin do Aisha gửi trong một lệnh gọi `send_message` và được đưa ra **theo thứ tự gửi** bởi Aisha. Mỗi phần tử của R là một mảng có độ dài 31, biểu diễn một gói tin bị sửa.
- Hàm này cần trả về một mảng có S bit bằng với thông điệp gốc M.
- Hàm này có thể được gọi **nhiều lần** trong mỗi trường hợp test, **chính xác một lần** cho mỗi lệnh gọi `send_message` tương ứng. **Thứ tự của các lệnh gọi hàm** `receive_message` không nhất thiết phải giống với thứ tự của các lệnh gọi `send_message` tương ứng.

Lưu ý rằng trong hệ thống chấm điểm, các thủ tục `send_message` và `receive_message` được gọi trong **hai chương trình riêng biệt**.

Các ràng buộc

- $1 \leq S \leq 1024$
- C có đúng 31 phần tử, trong đó 16 phần tử bằng 0 và 15 phần tử bằng $1.

Các Subtask và Chấm điểm

Nếu trong bất kỳ trường hợp test nào, các lệnh gọi đến hàm `send_packet` không tuân thủ các quy tắc được đề cập ở trên, hoặc giá trị trả về của bất kỳ lệnh gọi nào đến hàm `receive_message` là không chính xác, thì điểm cho lời giải của bạn đối với trường hợp test đó sẽ là 0.

Trái lại, gọi Q là số lượng lời gọi tối đa đến hàm `send_packet` trong số tất cả các lần gọi `send_message` trên tất cả các trường hợp test. Ngoài ra, đặt X bằng:

- 1, nếu $Q \leq 66$
- 0.95^{Q-66}, nếu $66 < Q \leq 100$
- 0, nếu $100 < Q$

Tiếp theo, điểm được tính như sau:

Subtask	Điểm	Ràng buộc thêm
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Không có ràng buộc gì thêm.

Lưu ý rằng trong một số trường hợp, hành vi của trình chấm điểm có thể là **thích ứng**. Điều này có nghĩa là các giá trị trả về bởi hàm `send_packet` có thể phụ thuộc vào không chỉ các tham số đầu vào của nó mà còn vào nhiều thứ khác, bao gồm các đầu vào và giá trị trả về của các lệnh gọi trước đó đến hàm này và các số giả ngẫu nhiên do trình chấm tạo ra. Trình chấm có tính **xác định** theo nghĩa là nếu bạn chạy nó hai lần và trong cả hai lần chạy, bạn gửi cùng một gói tin, nó sẽ thực hiện cùng một thay đổi đối với các gói tin đó.

Ví dụ

Xét lời gọi sau.

```
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])
```

Thông điệp mà Aisha cố gắng gửi tới Basma là $[0, 1, 1, 0]$. Các bit có chỉ số từ 0 đến 15 không thể bị Cleopatra thay đổi, trong khi các bit có chỉ số từ 16 đến 30 có thể được Cleopatra thay đổi.

Đối với ví dụ này, giả sử rằng Cleopatra sửa các bit liên tiếp mà cô ấy kiểm soát bằng cách điền xen kẽ 0 và 1, tức là cô ấy gán 0 đến chỉ số đầu tiên mà cô ấy kiểm soát (chỉ số 16 trong trường hợp của chúng ta), 1 tới chỉ số thứ hai mà cô ấy kiểm soát (chỉ số 17), 0 đến chỉ số thứ ba mà cô ấy kiểm soát (chỉ số 18), .v.v.

Aisha có thể quyết định gửi hai bit từ thông điệp gốc trong một gói tin như sau: cô ấy sẽ gửi bit đầu tiên tại 8 chỉ số đầu tiên do cô ấy kiểm soát và bit thứ hai ở 8 chỉ số tiếp theo do cô ấy kiểm soát.

Sau đó Aisha chọn gửi gói tin sau:

```
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])
```

Lưu ý rằng Cleopatra có thể thay đổi các bit với 15 chỉ số cuối cùng, vì vậy Aisha có thể thiết lập chúng một cách tùy ý vì chúng có thể bị ghi đè. Với chiến lược giả định của Cleopatra, hàm trả về: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Aisha quyết định gửi hai bit cuối cùng của M trong gói tin thứ hai theo cách tương tự như trên:

```
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])
```

Với chiến lược giả định của Cleopatra, hàm trả về: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Aisha có thể gửi nhiều gói tin nữa nhưng cô ta quyết định thôi.

Trình chấm tiếp theo gọi hàm sau:

```
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]])
```

Basma khôi phục thông điệp M như sau. Từ mỗi gói tin, cô ấy lấy bit đầu tiên xuất hiện hai lần liên tiếp, và bit cuối cùng xuất hiện hai lần liên tiếp. Nghĩa là, từ gói tin đầu tiên, cô ấy lấy các bit $[0,1]$ và từ gói tin thứ hai cô ấy lấy các bit $[1,0]$. Bằng cách ghép chúng lại với nhau, cô ấy khôi phục được thông điệp $[0,1,1,0]$, đây là giá trị trả về chính xác cho lệnh gọi `receive_message` này.

Có thể chứng minh rằng với chiến lược giả định của Cleopatra và đối với các thông điệp có độ dài 4, cách tiếp cận này của Basma khôi phục chính xác M, bất kể giá trị nào của C. Tuy nhiên, điều này không đúng trong trường hợp tổng quát.

Trình chấm mẫu

Trình chấm mẫu là không thích ứng. Thay vào đó, Cleopatra điền các bit liên tiếp mà cô ấy kiểm soát bằng các bit 0 và 1 xen kẽ nhau, như mô tả trong ví dụ trên.

Định dạng dữ liệu vào: **Dòng đầu tiên của đầu vào chứa một số nguyên T, là số lượng kịch bản.** Tiếp theo là T kịch bản. Mỗi kịch bản được cung cấp theo định dạng sau:

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

Định dạng kết quả ra: Trình chấm mẫu viết kết quả của mỗi kịch bản trong số T kịch bản theo cùng thứ tự như được cung cấp trong phần đầu vào theo định dạng sau:

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

Ở đây, K là số lượng lần gọi tới `send_packet`, D là thông điệp trả về bởi `receive_message` và L là độ dài thông điệp.