

## الرسالة

عائشة وبسمة صديقتان حميمتان.

تود عائشة إرسال رسالة  $M$ ، بطول  $S$  بت (بمعنى أصفار وواحدات)، إلى بسمة.

عائشة تتواصل مع بسمة من خلال **حزم** من البيانات.

الحزمة هي عبارة عن سلسلة مكونة من 31 بت مفهرسة من 0 إلى 30.

ترغب عائشة في إرسال الرسالة  $M$  إلى بسمة عن طريق إرسال عدد من الحزم.

ولكن لسوء الحظ، اخترقت كيلوباترا قناة الاتصال بين عائشة وبسمة، وتمكنت من **التلاعب** في الحزم المرسله عبر القناة.

حيث، أنه في كل حزمة بإمكان كيلوباترا التعديل في البتات بـ 15 موقع بالضبط.

بالتحديد، هناك مصفوفة تدعى  $C$  بطول 31، حيث أن كل عنصر في هذه المصفوفة يمكن أن يكون 0 أو 1، وفقاً للتفصيل الآتي:

•  $C[i] = 1$  يشير الى أن البت الذي موقعه  $i$  يمكن التلاعب به بواسطة كيلوباترا.

وتسمى هذه المواقع بـ **المواقع المسيطر عليها** بواسطة كيلوباترا.

•  $C[i] = 0$  يشير الى أن البت الذي موقعه  $i$  لا يمكن التلاعب به بواسطة كيلوباترا.

المصفوفة  $C$  تحوي بالتحديد 15 واحداً و 16 أصفاراً.

عند إرسال الرسالة  $M$  مجموعة المواقع المسيطر عليها بواسطة كيلوباترا تبقى نفسها بالنسبة لجميع الحزم.

عائشة تعرف بشكل دقيق من هي الـ 15 موقعاً المسيطر عليه بواسطة كيلوباترا.

ولكن بسمة تعرف فقط بوجود 15 موقع مسيطر عليه بواسطة كيلوباترا، ولكنها لا تعرف ما هي هذه المواقع بالضبط.

نفترض أن  $A$  تمثل الحزمة المراد إرسالها بواسطة عائشة (تسمى بـ **الحزمة الأصلية**)

نفترض أن  $B$  تمثل الحزمة التي استلمت بواسطة بسمة (تسمى بـ **الحزمة المتلاعب بها**)

لكل متغير  $i$ ، حيث أن  $0 \leq i < 31$ :

• إذا لم تسيطر كيلوباترا على البت الذي موقعه  $i$  ( $C[i] = 0$ )، في هذه الحالة فإن بسمة سوف تستقبل البت

الذي موقعه  $i$  كما أرسلته عائشة ( $B[i] = A[i]$ ).

• بخلاف ذلك، إذا سيطرت كيلوباترا على البت الذي موقعه  $i$  ( $C[i] = 1$ )، فإن قيمة  $B[i]$  ستحددها كيلوباترا.

مباشرة بعد إرسال كل حزمة، يمكن لعائشة معرفة الحزمة المتلاعب بها.

بعد أن تقوم عائشة بإرسال كل الحزم، في الطرف الآخر تستلم بسمه كل الحزم المتلاعب بها بنفس الترتيب الذي أرسلت به ويجب أن تكون قادرة على تكوين الرسالة الأصلية  $M$ .

يجب عليك تطوير وتنفيذ استراتيجية تسمح لعائشة بإرسال الرسالة  $M$  إلى بسمه، بحيث تكون بسمه قادرة على استرجاع الرسالة  $M$  من الحزم المتلاعب بها.

بالتحديد، عليك برمجة دالتين،

الدالة الأولى يستقبل الرسالة  $M$  والمصفوفة  $C$ ، ثم يقوم بإرسال مجموعة من الحزم إلى بسمه.

الدالة الثانية يمكن بسمه من معالجة الحزم المتلاعب بها واسترجاع الرسالة الأصلية  $M$ .

## تفاصيل البرمجة

الدالة الأولى:

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

- $M$ : مصفوفة بطول  $S$  تصف الرسالة التي تريد عائشة إرسالها إلى بسمه
- $C$ : مصفوفة بطول 31 تشير إلى مواقع (فهارس) البتات المسيطر عليها بواسطة كيوباترا
- يمكن استدعاء هذه الدالة على الأكثر 2100 مرة في كل حالة اختبار.

يجب عليك استدعاء الدالة بالطريقة التالية لإرسال كل حزمة.

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

- $A$ : الحزمة الأصلية (مصفوفة بطول 31) تمثل البتات المرسله من قبل عائشة.
- تعيد هذه الدالة الحزمة المتلاعب بها  $B$  تمثل البتات التي ستتسلمها بسمه.
- يمكنك استدعاء هذه الدالة 100 مرة على الأكثر في كل استدعاء لـ `send_message`.

الدالة الثاني:

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

- $R$ : مصفوفة تصف الحزم المتلاعب بها. تنشأ الحزم عند بسمه من الحزم المرسله من قبل عائشة عند كل عملية إرسال `send_message` بنفس الترتيب أرسلت به.

كل عنصر في المصفوفة  $R$  هو مصفوفة بطول 31، تمثل الحزم المتلاعب بها.

- يجب أن يعيد الدالة مصفوفة  $S$  بت والتي تساوي الرسالة الأصلية  $M$ .
- يمكن استدعاء هذه الدالة مرات عديدة في كل حالة اختبار و مرة واحدة لكل استدعاء موافق للدالة `send_message`. الترتيب `receive_message` استدعاءات الدالة ليست بالضرورة أن تكون بنفس الترتيب التي تمت فيها استدعاء الدالة `send_message`.

لاحظ أن الدالتين send\_message و receive\_message سيتم استدعاؤهما في برنامجين منفصلين.

## القيود

- $1 \leq S \leq 1024$
- $C$  تحوي بالضبط 31 عنصر، حيث يوجد 16 بت قيمتها 0 و 15 قيمتها 1.

## المسائل الجزئية والعلامات (الدرجات)

في أي حالة من حالات الاختبار، تم استدعاء الدالة send\_packet بطريقة لا تتوافق مع الشروط المشار إليها أعلاه أو كانت القيمة المرجعة من أي الاستدعاءات الدالة receive\_message غير صحيحة فإن العلامة لحلك في حالة الاختبار هذه ستكون 0.

وإلا ليكن  $Q$  هو أكبر عدد من الاستدعاءات الدالة send\_packet بين كل استدعاءات الدالة send\_message لكل حالات الاختبار. وليكن  $X$  يساوي:

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

عندئذ تحسب العلامة كما يلي:

المسألة الجزئية	النقاط	القيود الإضافية
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	.لا يوجد قيود إضافية

لاحظ أنه في بعض الحالات سيكون سلوك نظام التصحيح متكيفاً ذلك يعني أن القيم المعادة من الدالة send\_packet ربما لن تعتمد فقط على المدخلات ولكن أيضاً على العديد من الأشياء الأخرى، تتضمن الدخل والقيم المعادة من الاستدعاءات السابقة لهذه الدالة وبعض الأرقام العشوائية المولدة من قبل نظام التصحيح. أما نظام التصحيح فهو حتمي أي أنك إذا قمت بتشغيله مرتين وفي كل مرة قمت بإرسال نفس الحزم فإنه سيقوم بنفس التعديلات عليها..

## مثال

ليكن لدينا الاستدعاء التالي

```
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])
```

الرسالة التي ترغب عائشة بإرسالها إلى بسمه هي [0,1,1,0]. البتات ذات المواقع من 0 إلى 15 لا يمكن أن تتلاعب بها كيلوباترا، بينما البتات ذات المواقع من 16 حتى 30 يمكن أن يتغير محتواها من قبل كيلوباترا.

من أجل توضيح فكرة هذا المثال دعنا نفترض أن كيلوباترا تقوم بتعبئة البتات التي يمكنها التحكم بها بأصفار وواحدات بشكل متبادل على التوالي، ذلك يعني أنها تضع صفر في أول بت يمكنها التعديل عليه وهو البت 16 ثم تضع واحد في ثاني بت يمكنها التعديل عليه وهو البت رقم 17 ثم تضع صفر في ثالث بت يمكنها التعديل عليه وهو البت رقم 18 وهكذا

يمكن لعائشة أن تختار أن ترسل بتين من الرسالة الأصلية في حزمة واحدة كما يلي:

سترسل أول بت في أول 8 مواقع تتحكم بها وسترسل ثاني بت في المواقع الثمانية التالية التي تتحكم بها ثم تقوم عائشة بإرسال الحزمة التالية

```
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])
```

لاحظ أن كيلوباترا بإمكانها تغيير البتات في آخر 15 موقعاً لذلك فإن عائشة يمكنها وضع أي قيم عشوائية فيهم. حيث أنه من الممكن أن يتم التلاعب بهم. وبلاستراتيجية المفترضة لآلية عمل كيلوباترا يمكن أن تعيد الدالة البتات التالي:

$[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]$ . تقوم عائشة بإرسال البتين الأخيرين من الرسالة الأصلية  $M$  في الحزمة الثانية بنفس الطريقة السابقة.

```
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])
```

وفق الاستراتيجية المفترضة التي تتبعها كيلوباترا ستعيد الدالة ما يلي:

$[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]$  يمكن لعائشة أن ترسل حزم أكثر ولكن هي تختار أن لا تقوم بذلك

سيقوم نظام التصحيح بالاستدعاء التالي :

```
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, 0, 0, 0, 0, 0, 0, 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, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]])
```

تستعيد بسمة الرسالة  $M$  كما يلي: من كل حزمة تقوم بأخذ أول بت تتكرر مرتين متتاليتين وآخر بت يتكرر مرتين متتاليتين، أي أنها من الحزمة الأولى تأخذ البتين  $[0, 1]$ ، ومن الحزمة الثانية تأخذ البتين  $[1, 0]$ . وبعد وضعهم مع بعضهم ستتمكن من استعادة الرسالة الأصلية  $[0, 1, 1, 0]$ ، والتي ستشكل القيمة المعادة الصحيحة لاستدعاء الدالة `receive_message`.

من الممكن إثبات أن الاستراتيجية المفترضة لكيلوباترا ومن أجل رسائل بطول 4 بتات فإن هذه الاستراتيجية التي اتبعتها بسمة يمكنها استرجاع الرسالة  $M$  بغض النظر عن قيمة  $C$  ولكن هذا ليس صحيحاً من أجل الحالة العامة.

## Sample Grader

The sample grader is not adaptive. Instead, Cleopatra fills consecutive bits she controls with alternating 0 and 1 bits, as described in the example above

Input format: **The first line of the input contains an integer  $T$ , specifying the number of scenarios.**  $T$  scenarios follow. Each of them is provided in the following format

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

Output format: The sample grader writes the result of each of the  $T$  scenarios in the same order as they are provided in the input in the following format

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

Here,  $K$  is the number of calls to `send_packet`,  $D$  is the message returned by `receive_message` and  $L$  is its length