# Mensagem

Aisha e Basma são duas amigas que trocam mensagens entre si. Aisha tem uma mensagem $M$, que é uma sequência de $S$ bits (ou seja, zeros ou uns), que ela gostaria de enviar para Basma. Aisha se comunica com Basma enviando **pacotes**. Um pacote é uma sequência de $31$ bits indexados de $0$ a $30$ . Aisha gostaria de enviar a mensagem $M$ para Basma enviando-lhe um certo número de pacotes.

Infelizmente, Cleópatra comprometeu a comunicação entre Aisha e Basma e é capaz de **corromper** os pacotes. Ou seja, em cada pacote Cleópatra pode modificar bits em exatamente $15$ índices. Especificamente, existe um vetor $C$ de tamanho $31$, em que cada elemento é $0$ ou $1$, com o seguinte significado:

- $C[i] = 1$ indica que o bit com índice $i$ pode ser alterado por Cleópatra. Nós chamamos esses índices de **controlados** por Cleópatra.
- $C[i] = 0$ indica que o bit com índice $i$ não pode ser alterado por Cleópatra.

O vetor $C$ contém precisamente $15$ uns e $16$ zeros. Ao enviar a mensagem $M$, o conjunto de índices controlados por Cleópatra permanece o mesmo para todos os pacotes. Aisha sabe precisamente quais $15$ índices são controlados por Cleópatra. Basma só sabe que $15$ índices são controlados por Cleópatra, mas ela não sabe quais índices.

Seja $A$ um pacote que Aisha decide enviar (que chamamos de **pacote original**). Seja $B$ o pacote que é recebido por Basma (que chamamos de **pacote corrompido**). Para cada $i$, tal que $0 \leq i < 31$ :

- se Cleópatra não controla o bit com índice $i$ ($C[i] = 0$), Basma recebe o bit $i$ enviado por Aisha ($B[i] = A[i]$),
- caso contrário, se Cleópatra controla o bit com índice $i$ ($C[i] = 1$), o valor de $B[i]$ é decidido por Cleópatra.

Imediatamente após o envio de cada pacote, Aisha é informada sobre qual é o pacote corrompido correspondente.

Depois que Aisha envia todos os pacotes, Basma recebe todos os pacotes corrompidos **na ordem em que foram enviados** e tem que reconstruir a mensagem original $M$ .

Sua tarefa é elaborar e implementar uma estratégia que permita que Aisha envie a mensagem $M$ para Basma, de tal modo que Basma possa recuperar $M$ dos pacotes corrompidos.

Especificamente, você deve implementar dois procedimentos. O primeiro procedimento executa as ações de Aisha. O procedimento recebe uma mensagem $M$ e o vetor $C$, e deve enviar alguns pacotes para transferir a mensagem para Basma. O segundo procedimento executa as ações de Basma. Ele recebe os pacotes corrompidos e deve recuperar a mensagem original $M$.

## Detalhes de implementação

O primeiro procedimento que você deve implementar é:

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

- $M$: um vetor de tamanho $S$ descrevendo a mensagem que Aisha quer enviar a Basma.
- $C$: um vetor de tamanho $31$ indicando os índices de bits controlados por Cleópatra.
- Este procedimento pode ser chamado **no máximo 2100 vezes** em cada caso de teste.

Este procedimento deve chamar o seguinte procedimento para enviar um pacote:

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

- $A$: um pacote original (um vetor de tamanho $31$) representando os bits enviados por Aisha.
- Este procedimento retorna um pacote contaminado $B$ representando os bits que serão recebidos por Basma.
- Este procedimento pode ser chamado no máximo $100$ vezes em cada invocação de `send_message` .

O segundo procedimento que você deve implementar é:

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

- $R$: um vetor descrevendo os pacotes contaminados. Os pacotes são originários de pacotes enviados por Aisha em uma chamada `send_message` e são fornecidos **na ordem em que foram enviadas** por Aisha. Cada elemento de $R$ é um vetor de tamanho $31$, representando um pacote contaminado.
- Este procedimento deve retornar um vetor de $S$ bits que é igual à mensagem original $M$.
- Este procedimento pode ser chamado **várias vezes** em cada caso de teste, **exatamente uma vez** para cada chamada `send_message` correspondente. A **ordem** das **chamadas de procedimento** `receive_message` não é necessariamente o mesmo que a ordem das chamadas `send_message` correspondentes.

Observe que no corretor de exemplo os procedimentos `send_message` e `receive_message` são chamados em **dois programas separados**.

## Restrições

- $1 \leq S \leq 1024$
- $C$ tem exatamente $31$ elementos, dos quais $16$ são iguais a $0$ e $15$ são iguais a $1$.

## Subtarefas e pontuação

Se em qualquer um dos casos de teste, as chamadas ao procedimento `send_packet` não estão em conformidade com as regras mencionadas acima, ou o valor de retorno de qualquer uma das chamadas para o procedimento `receive_message` está incorreto, a pontuação da sua solução para esse caso de teste será $0$.

Caso contrário, seja $Q$ o número máximo de chamadas para o procedimento `send_packet` entre todas as invocações de `send_message` em todos os casos de teste. Adicionalmente, seja $X$ igual a:

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

Então, a pontuação é calculada da seguinte forma:

| Subtarefa | Pontuação | Restrições adicionais |
|---|---|---|
| 1 | $10 \cdot X$ | $S \leq 64$ |
| 2 | $90 \cdot X$ | Sem restrições adicionais. |

Observe que em alguns casos o comportamento do corretor pode ser **adaptativo**. Isso significa que os valores retornados por `send_packet` podem depender não apenas de seus argumentos de entrada, mas também de muitas outras coisas, incluindo as entradas e valores de retorno das chamadas anteriores para este procedimento e números pseudoaleatórios gerados pelo corretor. O corretor é **determinístico** no sentido que se você executá-lo duas vezes e em ambas as execuções você enviar os mesmos pacotes, ele fará as mesmas alterações neles.

## Exemplo

Considere a seguinte chamada.

```
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])
```

A mensagem que Aisha tenta enviar para Basma é $[0, 1, 1, 0]$. Os bits com índices de $0$ a $15$ não podem ser alterados por Cleópatra, enquanto os bits com índices de $16$ a $30$ podem ser alterados

por Cleópatra.

Para fins deste exemplo, vamos supor que o comportamento de Cleópatra preenche bits consecutivos que ela controla com $0$ e $1$ alternados, ou seja ela atribui $0$ para o primeiro índice que ela controla (índice $16$ no nosso caso), $1$ para o segundo índice que ela controla (índice $17$), $0$ para o terceiro índice que ela controla (índice $18$), e assim por diante.

Aisha pode decidir enviar dois bits da mensagem original em um pacote da seguinte maneira: ela enviará o primeiro bit nos primeiros $8$ índices que ela controla e o segundo bit nos índices $8$ seguintes ela controla.

Aisha então escolhe enviar o seguinte pacote:

```
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])
```

Observe que Cleópatra pode alterar bits com os últimos $15$ índices, para que Aisha possa defini-los arbitrariamente, pois eles podem ser sobrescritos. Com a estratégia assumida por Cleópatra, o procedimento retorna: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$ .

Aisha decide enviar os dois últimos bits de $M$ no segundo pacote de forma semelhante à anterior:

```
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])
```

Com a estratégia assumida por Cleópatra, o procedimento retorna: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$ .

Aisha pode enviar mais pacotes, mas ela decide não fazê-lo.

O corretor então faz a seguinte chamada de procedimento:

```
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 recupera a mensagem $M$ da seguinte forma. De cada pacote ela pega o primeiro bit que ocorre duas vezes seguidas, e o último bit que ocorre duas vezes seguidas. Ou seja, do primeiro pacote ela pega os bits $[0,1]$ e do segundo pacote ela pega os bits $[1,0]$. Ao juntá-los, ela recupera a mensagem $[0,1,1,0]$, que é o valor de retorno correto para esta chamada para `receive_message` .

Pode-se demonstrar que com a estratégia assumida por Cleópatra e para mensagens de tamanho $4$, esta abordagem de Basma recupera corretamente $M$, independentemente do valor de $C$. Contudo, isso não é correto no caso geral.

## Corretor de Exemplo

O corretor de exemplo não é adaptivol. Em vez disso, o comportamento de Cleópatra preenche bits consecutivos que ela controla com bits $0$ e $1$ alternados, conforme descrito no exemplo acima.

Formato de entrada: **A primeira linha da entrada contém um inteiro $T$, especificando o número de cenários.** Os cenários $T$ são os seguintes. Cada um deles é fornecido no seguinte formato:

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

Formato de saída: O corretor de exemplo escreve o resultado de cada um dos cenários $T$ na mesma ordem em que são fornecidos na entrada no seguinte formato:

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

Aqui, $K$ é o número de chamadas para `send_packet`, $D$ é a mensagem retornada por `receive_message` e $L$ é seu tamanho.