Message

A Aisha e a Basma são duas amigas que trocam correspondência entre si. A Aisha tem uma mensagem M, que é uma sequência de S bits (i.e., zeros ou uns), que ela gostaria de enviar à Basma. A Aisha comunica com a Basma ao enviar-lhe **packets**. Um packet é uma sequência de 31 bits indexados de 0 a 30. A Aisha gostaria de enviar uma mensagem M à Basma ao enviar-lhe algum número de packets.

Infelizmente, a Cleópatra comprometeu a comunicação entre a Aisha e a Basma e consegue **contaminar** os packets. Isto é, em cada packet a Cleópatra pode modificar bits em exatamente 15 índices. Especificamente, existe um array C de tamanho 31, no qual todos os elementos são 0 ou 1, com o seguinte significado:

- $C[i] = 1$ indica que o bit com índice i pode ser mudado pela Cleópatra. Chamamos a estes índices **controlados** pela Cleópatra.
- $C[i] = 0$ indica que o bit com índice i não pode ser mudado pela Cleópatra.

O array C contém precisamente 15 uns e 16 zeros. Enquanto a mensagem M é enviada, o conjunto de índices controlados pela Cleópatra mantém-se o mesmo para todos os packets. A Aisha sabe precisamente quais 15 índices são controlados pela Cleópatra. A Basha apenas sabe que 15 índices são controlados pela Cleópatra, mas não sabe quais índices são.

Seja A um packet que a Aisha decide enviar (a que chamamos **packet original**). Seja B o packet recebido pela Basma (a que chamamos **packet contaminado**). Para cada i, tal que $0 \leq i < 31$:

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

Imediatamente após enviar cada packet, a Aisha fica a saber qual o packet contaminado correspondente.

Depois da Aisha enviar todos os packets, a Basma recebe todos os packets contaminados **na ordem em que foram enviados** e tem de reconstruir a mensagem original M.

A tua tarefa é planear e implementar uma estratégia que permita à Aisha enviar a mensagem M à Basma, de forma a que a Basma consiga recuperar M dos packets contaminados. Especificamente, deves implementar duas funções. A primeira função faz as ações da Aisha.

Recebe a mensagem M e o array C, e envia alguns packets para transferir a mensagem à Basma. A segunda função faz as ações da Basma. Recebe os packets contaminados e deve recuperar a mensagem original M.

Detalhes de Implementação

A primeira função que deves implementar é:

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

- M: um array de tamanho S descrevendo a mensagem que a Aisha quer enviar à Basma.
- C: um array de tamanho 31 indicando os índices dos bits controlados pela Cleópatra.
- Esta função deve ser chamada **no máximo 2100 vezes** em cada caso de teste.

Esta função deve chamar a seguinte função para enviar um packet:

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

- A: um packet original (um array de tamanho 31) representando os bits enviados pela Aisha.
- Esta função retorna um packet contaminado B representando os bits que vão ser recebidos pela Basma.
- Esta função pode ser chamada no máximo 100 times em cada invocação de `send_message`.

A segunda função que deves implementar é:

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

- R: um array a descrever os packets contaminados. Os packets são originados de packets enviados pela Aisha numa invocação de `send_message` e são dados **na ordem em que foram enviados** pela Aisha. Cada elemento de R é um array de tamanho 31, representando um packet contaminado.
- Esta função deve retornar um array de S bits que é igual à mensagem original M.
- Esta função pode ser chamada **multiplas vezes** em cada caso de teste, **exatamente uma vez** para cada correspondente chamada a `send_message`. A **ordem das chamadas a** `receive_message` não é necessariamente a mesma da ordem das correspondentes chamadas a `send_message`.

Nota que no sistema de avaliação as funções `send_message` e `receive_message` são chamadas 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 dos casos de teste, as chamadas à função `send_packet` não forem conforme as regras acima mencionadas, ou o valor devolvido nalguma das chamadas à função `receive_message` for incorreto, a pontuação da tua solução nesse caso de teste será 0.

Por outro lado, seja Q o máximo número de chamadas à função `send_packet` entre todas as chamadas a `send_message` em todos os casos de teste. Seja também X igual a:

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

Então, a pontuação final é calculada da seguinte forma:

Subtarefa	Pontos	Restrições Adicionais
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Sem restrições adicionais.

Nota que em alguns casos o comportamento do avaliador pode ser **adaptativo**. Isto significa que os valores devolvidos por `send_packet` podem depender não só dos seus argumentos de input e do valor devolvido em chamadas anteriores a esta função.

Nota que em alguns casos o comportamento do avaliador pode ser **adaptativo**. Isto significa que os valores devolvidos por `send_packet` podem depender não só dos seus argumentos de input mas também de várias outras coisas, incluindo os inputs e valores devolvidos em chamadas anteriores a esta função e números pseudo-aleatórios gerados pelo avaliador. O avaliador é **determinístico** no sentido em que se o correres duas vezes e em ambas as vezes enviares os mesmos packets, este fará as mesmas alterações neles.

Exemplo

Considera 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 a Aisha tenta enviar à Basma é $[0, 1, 1, 0]$. Os bits com índices de 0 a 15 não podem ser mudados pela Cleópatra, enquanto que os bits com índices de 16 a 30 podem ser mudados pela Cleópatra.

Para este exemplo, vamos assumir que a Cleópatra enche bits consecutivos que controla com 0 e 1 alternadamente, i.e. ela atribui 0 ao primeiro índice que controla (índice 16 no nosso caso), 1 ao segundo índice que controla (índice 17), 0 ao terceiro índice que controla (índice 18), e por aí adiante.

A Aisha pode decidir enviar dois bits da mensagem original num packet da seguinte maneira: ela vai enviar o primeiro bit nos primeiros 8 índices que controla e o segundo bit nos seguintes 8 índices que ela controla.

A Aisha decide então enviar o seguinte packet:

```
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])
```

Nota que a Cleópatra pode mudar bits com os últimos 15 índices, portanto a Aisha pode coloca-los arbitrariamente, visto que podem ser alterados. Com a estratégia assumida da Cleópatra, a função retorna: $[0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

A Aisha decide enviar os últimos dois bits de M no segundo packet numa maneira 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 da Cleópatra, a função retorna: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

A Aisha pode enviar mais packets, mas escolhe não o fazer.

Depois, o avaliador faz a seguinte chamada:

```
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]])
```

A Basma recupera a mensagem M da seguinte maneira: de cada packet ela tira o primeiro bit que ocorre duas vezes consecutivas e o último bit que ocorre duas vezes consecutivas. Isto é, do primeiro packet, ela tira os bits $[0,1]$, e do segundo packet ela tira $[1,0]$. Ao juntar estes bits, ela recupera a mensagem $[0,1,1,0]$, que é o valor correto a devolver para esta chamada de `receive_message`.

Pode ser mostrado que, com a estratégia assumida da Cleópatra e para mensagens de tamanho 4, esta abordagem da Basma recupera corretamente M, independentemente do valor de C. No entanto, não está correta no caso geral.

Avaliador Exemplo

O avaliador de exemplo não é adaptativo. Ao invés, a Cleópatra enche bits consecutivos que controla com 0 e 1 alternadamente, como descrito no exemplo acima.

Formato de input: **A primeira linha do input contém um inteiro T, especificando o número de cenários.** Seguem T cenários. Cada um dos cenários é dado no seguinte formato:

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

Formato de output: O avaliador exemplo escreve o resultado de cada um dos T cenários na mesma ordem em que foram dados no input, no seguinte formato:

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

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