Mensaje

Aisha y Basma son dos amigas que se envían mensajes la una a la otra. Aisha tiene un mensaje M para Basma, el cual es una secuencia de S bits (es decir, ceros o unos) que desea enviarle a Basma. Aisha se comunica con Basma enviándole **paquetes**. Un paquete es una secuencia de 31 bits indexados desde 0 hasta 30. A Aisha le gustaría enviarle el mensaje M a Basma enviándole cierto número de paquetes.

Desafortunadamente, Cleopatra interceptó la comunicación entre Aisha y Basma y es capaz de **modificar** los paquetes. Es decir, en cada paquete Cleopatra puede modificar bits en exactamente 15 índices. Específicamente, hay un arreglo C de largo 31 en el cual cada elemento tiene el valor de 0 o 1, que puede interpretarse de la siguiente manera:

- $C[i] = 1$ Indica que el bit con indice i puede ser cambiado por Cleopatra. Llamamos a estos índices como **controlados** por Cleopatra.
- $C[i] = 0$ Indica que el bit con índice i no puede ser cambiado por Cleopatra.

El arreglo C contiene exactamente 15 unos y 16 ceros. Mientras se envía el mensaje M, el conjunto de índices controlados por Cleopatra se mantiene igual para todos los paquetes. Aisha conoce exactamente cuáles 15 índices son controlados por Cleopatra. Basma sólo sabe que 15 índices son controlados por Cleopatra, pero no sabe cuáles son dichos índices.

Sea A un paquete que Aisha decide enviar (al cual le llamaremos el **paquete original**). Sea B el paquete que es recibido por Basma (al cual le llamaremos el **paquete modificado**). Para cada i, tal que $0 \leq i < 31$:

- Si Cleopatra no controla el bit con índice i ($C[i] = 0$), Basma recibe el bit en el índice i exáctamente igual a como lo envió Aisha ($B[i] = A[i]$),
- En caso contrario, si Cleopatra controla el bit con el índice i ($C[i] = 1$), entonces ella decide el valor de $B[i]$.

Inmediatamente después de enviar cada paquete, Aisha conoce cuál el contenido del paquete modificado.

Después que Aisha envía todos los paquetes, Basma recibe todos los paquetes modificados **en el orden en el que han sido enviados** y tiene que reconstruir el mensaje original M.

Tu tarea es idear e implementar una estrategia con la cual Aisha le mande el mensaje M a Basma, de tal manera que Basma logre recuperar M a partir de los paquetes modificados.

Específicamente, deberás implementar dos funciones: La primera función simula las acciones de Aisha. Recibe un mensaje M y el arreglo C, y deberá enviar algunos paquetes para transmitirle el mensaje a Basma. La segunda función realiza las acciones de Basma. Se le entregan los paquetes modificados y deberá recuperar el mensaje original M.

Detalles de implementación

La primera función que debes implementar es la siguiente:

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

- M: un arreglo de largo 31 con el mensaje que Aisha quiere enviarle a Basma.
- C: un arreglo de largo 31 que indica los índices de los bits controlados por Clepoatra.
- esta función puede ser llamada **a lo sumo 2100 veces** en cada caso de prueba.

La función `send_message` deberá llamar a la siguiente función para enviar un paquete:

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

- A: uno de los paquetes originales (un arreglo de largo 31) representando los bits enviados por Aisha.
- Esta función retorna un mensaje modificado B representando los bits que recibirá Basma.
- Esta función puede ser llamada a lo sumo 100 veces en cada invocación de `send_message`.

La segunda función que debes implementar es la siguiente:

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

- R: un arreglo describiendo los paquetes modificados. Es decir, contiene los paquetes provenientes de los enviados por Aisha en una llamada a `send_message` y son dados **en el orden en el que fueron enviados** por Aisha. Cada elemento de R es un arreglo de tamaño 31, representando un paquete modificado.
- Esta función deberá retornar un arreglo de S bits que deberá ser igual al mensaje original M.
- Esta función puede ser llamada **múltiples veces** en cada caso de prueba, **exactamente una vez** por cada llamada a `send_message`.

El **orden de las llamadas a la función** `receive_message` no es necesariamente el mismo orden de llamada a las funciones `send_message`.

Nota que en el sistema calificador las funciones de `send_message` y `receive_message` son llamadas en **dos programas diferentes**.

Restricciones

- $1 \leq S \leq 1024$
- C tiene exactamente 31 elementos, de los cuales 16 son iguales a 0 y 15 son iguales a 1.

Subtareas y puntaje

Si en cualquiera de los casos de prueba, las llamadas a la función `send_packet` no satisfacen las reglas mencionadas anteriormente, o el valor que retorna cualquiera de las llamadas a la función `receive_message` es incorrecto, el puntaje de tu solución para ese caso será 0.

En cualquier otro caso, sea Q el máximo número de llamadas a la función `send_packet` entre todas las llamadas a `send_message` a través de todos los casos de prueba. También, sea X igual a:

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

Entonces el puntaje se calcula como se describe a continuación:

Subtarea	Puntaje	Restricciones adicionales
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Sin restricciones adicionales

Nota que en algunos casos el comportamiento del calificador puede ser **adaptativo**. Esto significa que los valores retornados por `send_packet` pueden depender no sólo de los parámetros de entrada, sino también en muchas otras cosas, incluyendo las entradas y los valores de retorno de llamadas previas a esta función y números pseudo-aleatorios generados por el calificador. El calificador es **determinista** en el sentido que si lo ejecutas dos veces y en ambos casos envías los mismos paquetes, hará los mismos cambios a los mismos.

Ejemplo

Considera la siguiente llamada.

```
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])
```

El mensaje que Aisha le intenta enviar a Basma es $[0, 1, 1, 0]$. Los bits con indices desde 0 hasta 15 no pueden ser modificados por Cleopatra, mientras que los indices desde 16 hasta 30 sí pueden ser cambiados por Cleopatra.

Para este ejemplo, asumamos que Cleopatra modifica bits consecutivos que ella controla alternando entre 0 y 1; es decir, que ella asigna 0 al primer índice que ella controla (índice 16 in nuestro caso), 1 al segundo índice que ella controla (índice 17), 0 al tercer índice que ella controla (índice 18), y así sucesivamente.

Aisha puede decidir enviar dos bits desde el mensaje original en un paquete de la siguiente manera: ella enviará el primer bit en los primeros 8 indices que ella controla y el segundo bit en los siguientes 8 índices que ella controla.

Entonces, Aisha envía el siguiente paquete:

```
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 Cleopatra puede cambiar los bits de los últimos 15 índices, así que Aisha puede asignarlos arbitrariamente, ya que luego podrán ser sobreescritos. Con la estrategia que asumimos de Cleopatra, la función retornará: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Aisha decide enviar los últimos dos digitos de M en el segundo paquete de una manera similar a la 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])
```

Haciendo uso de la estrategia asumida por Cleopatra, la función retorna lo siguiente: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Aisha puede enviar más paquetes, pero ella escoje no seguir enviando.

Entonces, el calificador hace la siguiente llamada:

```
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 el mensaje M de la siguiente manera: Para cada paquete, ella toma el primer bit que aparece dos veces seguidas, y el último bit que aparece dos veces seguidas. Es decir, para el primer paquete, ella toma los bits $[0,1]$, y para el segundo paquete ella toma los bits $[1,0]$. Al ponerlos juntos, ella recupera el mensaje $[0,1,1,0]$, el cual es el valor correcto de retorno para esta llamada a `receive_message`.

Se puede demostrar que con la estrategia asumida de Cleopatra y para mensajes de largo 4, esta estrategia de Basma recupera correctamente a M, sin importar el valor de C. Sin embargo, no es correcta para el caso general.

Calificador local

El calificador local no es adaptativo. En cambio, Cleopatra modifica bits consecutivos que ella controla alternando los bits entre 0 y 1 como en el ejemplo explicado anteriormente.

Formato de entrada: **La primera linea de entrada contiene un entero T, especificando el número de escenarios.** T escenarios le siguen. Cada uno de ellos se presenta con el siguiente formato:

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

Formato de salida: El calificador local escribe el resultado de cada uno de los T escenarios en el mismo orden en el que se ingresan en la entrada utilizando siguiente formato:

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

Donde, K es el número de llamadas a `send_packet`, D es el mensaje retornado por `receive_message` y L es su longitud.