Mechanical Doll

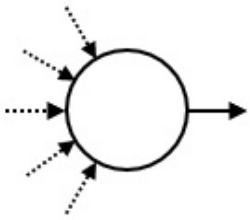
Una muñeca mecánica es una muñeca que repite automáticamente una secuencia específica de movimientos. En Japón se han creado muchas muñecas mecánicas desde la antigüedad.

Los movimientos de una muñeca mecánica son controlados por un **circuito** que está formado por **dispositivos**. Los dispositivos están conectados mediante tubos. Cada dispositivo tiene una o dos **salidas**, y puede tener una cantidad arbitraria de **entradas** (posiblemente cero). Cada tubo conecta una salida de un dispositivo con una entrada de un dispositivo (posiblemente el mismo). Cada salida se encuentra conectada con exactamente un tubo, y cada entrada se encuentra conectada con exactamente un tubo.

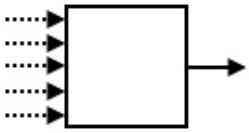
Para describir los movimientos de la muñeca, supongamos que se coloca una **bolita** en uno de los dispositivos. La bolita viaja a través del circuito. En cada paso de su recorrido, la bolita sale del dispositivo por una de sus salidas, viaja a lo largo del tubo conectado a esa salida, y entra al dispositivo que se encuentra del otro lado del tubo.

Existen tres tipos de dispositivos, denominados **origin**, **trigger**, y **switch**. Hay exactamente un origin, M triggers, y S switches (S puede ser cero). Tu programa debe decidir el valor de S. Cada dispositivo tiene un número de serie único.

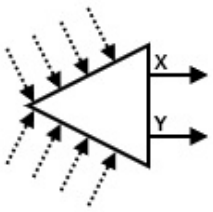
La bolita comienza su recorrido en el dispositivo origin. Este dispositivo tiene exactamente una salida. Su número de serie es 0.



Un dispositivo trigger provoca un movimiento específico de la muñeca cada vez que la bolita entra al mismo. Cada trigger tiene exactamente una salida. Los números de serie de los triggers van desde 1 hasta M.



Cada switch tiene exactamente dos salidas, denominadas 'X' e 'Y'. El **estado** de un switch puede ser o bien 'X', o bien 'Y'. Luego de entrar a un cierto switch, la bolita sale del mismo utilizando la salida correspondiente al estado actual del switch. Luego de eso, el switch cambia su estado al estado contrario. En un comienzo, el estado de todos los switches es 'X'. Los números de serie de los switches van desde -1 hasta $-S$.



Tu programa recibe la cantidad de triggers M. También recibe una secuencia A de longitud N, cuyos elementos son números de serie de dispositivos de tipo trigger. Cada trigger puede aparecer varias veces en la secuencia A (incluso cero veces). Tu programa debe crear un circuito que satisfaga las siguientes condiciones:

- La bolita vuelve al dispositivo origin luego de cierta cantidad de pasos.
- Cuando la bolita vuelve al dispositivo origin por primera vez, el estado de cada switch es 'X'.
- La bolita vuelve al dispositivo origin por primera vez luego de haber entrado a dispositivos de tipo trigger exactamente N veces. Los números de serie de los dispositivos de tipo trigger, en el orden en que la bolita ingresó a ellos, son $A_0, A_1, \ldots, A_{N-1}$.
- Sea P la cantidad total de cambios de estado de todos los switches causados por la bolita antes de su primer retorno al dispositivo origin. El valor de P no excede $20\,000\,000$.

A su vez, no es deseable utilizar muchos switches.

Detalles de implementación

Debes implementar la siguiente función:

```
create_circuit(int M, int[] A)
```

- `M`: la cantidad de triggers.
- `A`: un arreglo de longitud N, con los números de serie de los dispositivos de tipo trigger en los cuales la bolita debe entrar, en el orden en que debe recorrerlos.
- Esta función es llamada exactamente una vez.
- Notar que el valor de N es la longitud del arreglo `A`, y puede obtenerse como lo indican las notas de implementación.

Para dar la respuesta, se debe llamar a la siguiente función:

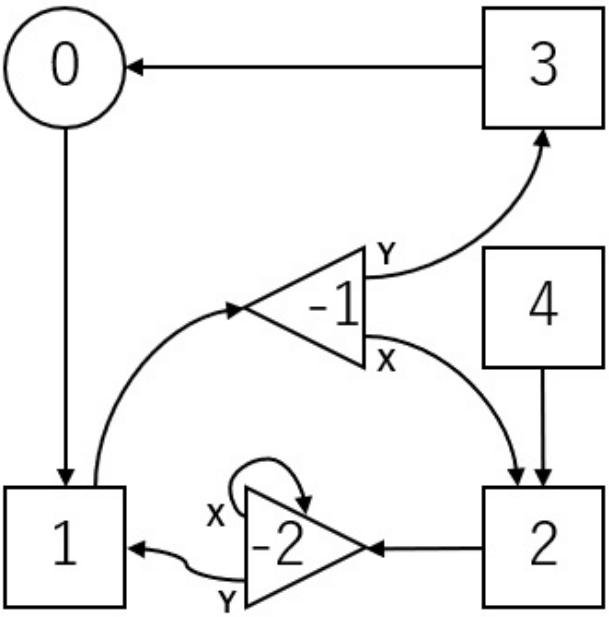
```
answer(int[] C, int[] X, int[] Y)
```

- `C`: un arreglo de longitud $M+1$. La salida del dispositivo i ($0 \leq i \leq M$) está conectada con el dispositivo `C[i]`.
- `X`, `Y`: arreglos de la misma longitud. La longitud S de estos arreglos es la cantidad de switches. Para el switch $-j$ ($1 \leq j \leq S$), su salida 'X' está conectada con el dispositivo `X[j - 1]` y su salida 'Y' está conectada con el dispositivo `Y[j - 1]`.
- Cada uno de los elementos de `C`, `X`, e `Y` debe ser un entero entre $-S$ y M, inclusive.
- S debe ser como máximo $400\,000$.
- Esta función debe ser llamada exactamente una vez.
- El circuito representado por los arreglos `C`, `X`, e `Y` debe satisfacer las condiciones del enunciado.

Si no se satisface alguna de las condiciones anteriores, el programa será juzgado como **`Wrong Answer`**. De lo contrario, el programa será juzgado como **`Accepted`** y se calculará el puntaje a partir del valor de S (ver Subtareas).

Ejemplo

Sea $M = 4$, $N = 4$, y $A = [1, 2, 1, 3]$. El grader realiza la llamada `create_circuit(4, [1, 2, 1, 3])`.



La figura muestra un circuito, correspondiente a la llamada `answer([1, -1, -2, 0, 2], [2, -2], [3, 1])`. Los números que se ven en la figura son los números de serie de los dispositivos.

Se utilizan dos dispositivos de tipo switch, y por lo tanto $S = 2$.

En un comienzo, los estados de los switches -1 y -2 son ambos 'X'.

La bolita realiza el siguiente recorrido:

$$0 \longrightarrow 1 \longrightarrow -1 \xrightarrow{\text{X}} 2 \longrightarrow -2 \xrightarrow{\text{X}} -2 \xrightarrow{\text{Y}} 1 \longrightarrow -1 \xrightarrow{\text{Y}} 3 \longrightarrow 0$$

- Cuando la bolita entra por primera vez al switch -1, su estado es 'X'. Por lo tanto, la bolita viaja al dispositivo 2. Además el estado del switch -1 cambia y pasa a ser 'Y'.
- Cuando la bolita entra por segunda vez al switch -1, su estado es 'Y'. Por lo tanto, la bolita viaja al dispositivo 3. Además el estado del switch -1 cambia y pasa a ser 'X'.

La bolita vuelve por primera vez al dispositivo origin, habiendo entrado a los triggers $1, 2, 1, 3$. Los estados de los switches -1 y -2 son ambos 'X'. El valor de P es 4. Por lo tanto, este circuito satisface las condiciones.

El archivo `sample-01-in.txt` en el zip adjunto se corresponde con este ejemplo. Hay también otros ejemplos disponibles en el zip.

Restricciones

- $1 \leq M \leq 100\,000$
- $1 \leq N \leq 200\,000$
- $1 \leq A_k \leq M$ $(0 \leq k \leq N-1)$

Subtareas

A continuación se muestra el puntaje y las restricciones para cada caso de prueba:

1. (2 puntos) Para cada i $(1 \leq i \leq M)$, el entero i aparece a lo sumo una vez en la secuencia $A_0, A_1, \ldots, A_{N-1}$.
2. (4 puntos) Para cada i $(1 \leq i \leq M)$, el entero i aparece a lo sumo dos veces en la secuencia $A_0, A_1, \ldots, A_{N-1}$.
3. (10 puntos) Para cada i $(1 \leq i \leq M)$, el entero i aparece a lo sumo 4 veces en la secuencia $A_0, A_1, \ldots, A_{N-1}$.
4. (10 puntos) $N = 16$
5. (18 puntos) $M = 1$
6. (56 puntos) Sin restricciones adicionales

Para cada caso de prueba en el que el programa sea juzgado como **`Accepted`**, se calcula el puntaje a partir del valor de S:

- Si $S \leq N + \log_2 N$, se obtienen todos los puntos del caso de prueba.
- Para cada caso de prueba correspondiente a las Subtareas 5 y 6, si $N + \log_2 N < S \leq 2N$, se obtiene puntaje parcial. El puntaje del caso de prueba

es $0.5 + 0.4 \times \left(\frac{2N - S}{N - \log_2 N} \right)^2$, multiplicado por el puntaje correspondiente a la subtarea.
- De lo contrario, se obtienen cero puntos.

Nota que el puntaje obtenido en una subtarea es el mínimo de los puntajes obtenidos en los casos de prueba de dicha subtarea.

Grader de ejemplo

El grader de ejemplo lee la entrada estándar con el siguiente formato:

- línea 1: M N
- línea 2: A_0 A_1 $\ldots$ A_{N-1}

El grader de ejemplo genera tres salidas.

En primer lugar, el grader de ejemplo escribe la respuesta del programa en un archivo llamado `out.txt` con el siguiente formato:

- línea 1: S
- línea $2 + i$ ($0 \le i \le M$): `C[i]`
- línea $2 + M + j$ ($1 \le j \le S$): `X[j - 1] Y[j - 1]`

En segundo lugar, el grader de ejemplo simula el recorrido de la bolita. Escribe los números de serie de los dispositivos en los que la bolita entró, en orden, en un archivo llamado `log.txt`.

En tercer lugar, el grader de ejemplo escribe a la salida estándar la evaluación del resultado del programa.

- Si el programa es juzgado como **`Accepted`**, el grader escribe S y P con el siguiente formato `Accepted: S P`.

- Si el programa es juzgado como **`Wrong Answer`**, escribe `Wrong Answer: MSG`. El significado de `MSG` es el siguiente:

  - `answered not exactly once`: La función `answer` no fue llamada exactamente una vez.
  - `wrong array length`: La longitud de `C` no es $M + 1$, o las longitudes de `X` e `Y` son diferentes.
  - `over 400000 switches`: S es mayor que $400\,000$.
  - `wrong serial number`: Hay un elemento de `C`, `X`, o `Y` que es menor que $-S$ o mayor que M.
  - `over 20000000 inversions`: La bolita no retorna al dispositivo origin tras $20\,000\,000$ de cambios de estado de los switches.
  - `state 'Y'`: Existe un switch cuyo estado es 'Y' en el momento en que la

bolita retorna por primera vez al dispositivo origin.
- `wrong motion`: Los dispositivos de tipo trigger recorridos no se corresponden con la secuencia A.

Notar que el grader de ejemplo podría no crear `out.txt` y/o `log.txt` cuando el programa es juzgado como `Wrong Answer`.