D. Decoraciones de Jardín

Nombre del problema	Decoraciones de Jardín
Límite de Tiempo	7 segundos
Límite de Memoria	1 gigabyte

Cada día, cuando va al colegio y durante la vuelta a casa, Detje pasa por una calle con N casas, numeradas desde el 0 hasta $N-1$. Actualmente, la casa i está habitada por la persona i. Para un cambio de paisaje, los residentes han decidido de intercambiar casas entre ellos. La persona que se mudará a la casa i es la persona a_i (que en este momento vive en la casa a_i).

Cada casa tiene la estatua de un pájaro en el jardín. Las estatuas tienen dos estados posibles: sus alas están o bien *abiertas* (como si el pájaro volara) o *cerradas* (como si el pájaro estuviera plantado sobre el suelo). Los residentes tienen fuertes preferencias respecto a la apariencia de sus estátuas de pájaro. Actualmente, el pájaro enfrente de la casa i se encuentra en la configuración preferida por el residente i. Los residentes se niegan a mudarse a una casa si su estátua no se encuentra en su configuración preferida. Detje quiere ayudarles a organizar las estatuas de manera que se puedan mudar.

Con ese propósito, reliza lo siguiente: cuando pasa por la calle (o bien en el camino a la escuela o de vuelta a casa), observa los pájaros que pasa uno por uno, y posiblemente ajusta algunas de las estatuas (abriendo o cerrando las alas). Dado que sus días en el colegio y en casa son muy ocupados, **Detje no recuerda los estados de los pájaros que vió en sus paseos anteriores**. Afortunadamente, ha apuntado la lista $a_0, a_1, ..., a_{N-1}$, de manra que sabe qué residente se muda a dónde.

Ayuda a Detje a diseñar una estrategia que le indique qué pájaros manipular con tal de ajustar las estatuas a las preferencias de los esidentes. Detje puede pasar por la calle como máximo 60 veces, pero para obtener una puntuación mayor deberá pasar por la calle menos veces.

Implementación

Este es un problema multiejecución, lo que significa que tu programa se ejecutará múltiples veces.

En cada ejecución, primero deberás leer una línea con dos enteros w y N, el índice del paseo y el número de casas. En la primera ejecución de tu programa $w = 0$, en la segunda $w = 1$, y así

continúa (más tarde se explican más detalles).

En la segunda línea de entrada se encuentran N enteros $a_0, a_1, ..., a_{N-1}$, indicando que la persona que se mudará a la casa i se encuentra viviendo en la casa a_i. Los a_is forman una *permutación*: cada número desde el 0 hasta $N-1$ aparece exactamente una vez en la lista de a_is. Observad que un residente puede elegir no moverse: es decir, se permite $a_i = i$.

Los residentes solo intercambiarán casas una vez. Eso quiere decir que para un caso de prueba fijo, el valor de N y la lista de a_is será la misma para todas las ejecuciones de tu programa.

Primera ejecución

Para la primera ejecución de tu programa, $w = 0$. En esta ejecución, únicamente deberás imprimir un entero W ($0 \leq W \leq 60$), el número de veces que quieres que Detje pase por las casas. Entonces tu programa deberá finalizar. A continuación, tu programa será ejecutado W veces más.

Ejecuciones subsecuentes

En la siguiente ejecución de tu programa, $w = 1$; en la siguiente tras esa $w = 2$; y así hasta la última ejecución donde $w = W$.

Una vez hayas leído w, N y los $a_0, a_1, ..., a_{N-1}$, Detje empieza a caminar por la calle.

- Si w es impar, Detje camina desde su casa hasta la escuela, y pasa las casas en el orden $0, 1, ..., N-1$.

  Tu programa deberá ahora leer una línea que contiene b_0, o bien 0 (cerrado) o 1 (abierto), el estado actual de la estatua en frente de la casa 0. Ttas leer b_0, deberás imprimir una línea con 0 o 1, el nuevo valor que quieras que tenga b_0.

  Entonces tu programa deberá leer una línea con b_1, el estado de la estatua en frente de la casa 1; y posteriormente imprimir el nuevo valor de b_1. Esto continúa por cada una de las N casas. Una vez Detje passe la última casa (es decir, una vez lea y escriba b_{N-1}), **tu programa deberá finalizar**.

  Observad que el programa solo puede leer el siguiente valor b_{i+1} una vez haya escrito el valor de b_i.

- Si w es par, Detje camina desde la escuela hasta su casa, y pasará las casas en el orden inverso $N-1, N-2, ..., 0$.

  El proceso es el mismo que cuando w es impar, excepto porque empiezas leyendo y escribiendo b_{N-1}, luego b_{N-2}, y así hasta b_0.

Cuando $w = 1$, los valores de entrada $b_0, b_1, ..., b_{N-1}$ son los estados originales de las estatuas de pájaros (que son también las configuraciones preferidas por los residentes). Cuando $w > 1$, los valores de entrada $b_0, b_1, ..., b_{N-1}$ serán los que la ejecución previa de tu programa les asignó.

Al final, tras la última ejecución de tu programa, el valor de b_i debe ser igual al valor original de b_{a_i} para todo i, de lo contrario obtendrás el veredicto `Wrong Answer`.

Detalles

Si la *suma* de los tiempos de ejecución de las $W + 1$ ejecuciones independientes de tu programa excede el tiempo límite, tu envío será juzgado como `Time Limit Exceeded`.

Asegúrate de hacer *flush* de tu salida tras imprimir cada línea, o de lo contrario tu programa puede ser juzgado como `Time Limit Exceeded`. En Python, esto sucede automáticamente siempre y cuando uses `input()` para leer líneas. En C++, `cout << endl;` hace *flush* aparte de imprimir una nueva línea; si usas printf, utiliza `fflush(stdout)`.

Restricciones y puntuación

- $2 \leq N \leq 500$.
- Puedes utilizar como máximo $W \leq 60$ rondas.

Tu solución será evaluada en un conjunto de subtareas, cada una valorada en un número de puntos. Cada subtarea contendrá un número de casos de prueba. Para obtener los puntos de una subtarea, debes resolver todos los casos de prueba en la subtarea.

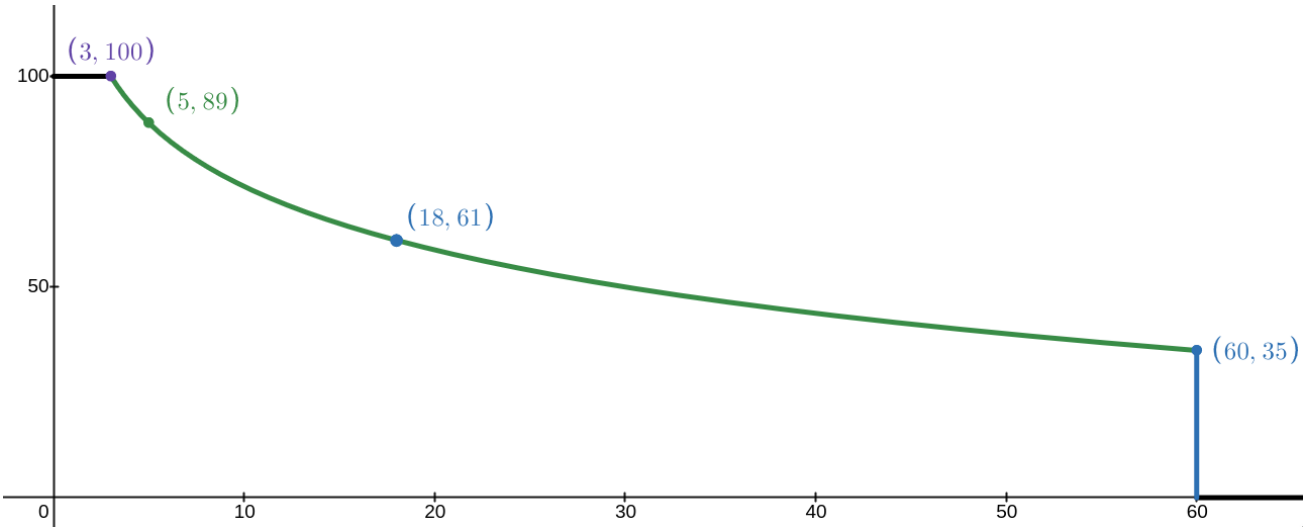
Subtarea	Puntuación máxima	Restricciones
1	10	$N = 2$
2	24	$N \leq 15$
3	9	$a_i = N - 1 - i$
4	13	$a_i = (i + 1) \bmod N$
5	13	$a_i = (i - 1) \bmod N$
6	31	Sin restricciones adicionales

Para cada subtarea que tu programa resuelva correctamente, recibirá una puntuación basada en la siguiente fórmula:

$$\text{score} = S_g \cdot \left(1 - \tfrac{1}{2} \log_{10}(\max(W_g, 3)/3)\right),$$

donde S_g es la máxima puntuación para la subtarea, y W_g es el máximo valor de W empleado para cualquier caso de prueba de la subtarea. Tu puntuación en cada subtarea se redondeará al entero más cercano.

La gráfica muestra el número de puntos, como función de W, que tu programa obtendrá si resuelves todas las subtareas con el mismo valor de W. En particular, para obtener una puntuación de 100 puntos en este problema, deberás resolver cada caso de prueba con $W \leq 3$.



Herramienta de testing

Para facilitar el testing de tu solución, facilitamos una herramienta simple que puedes descargar. Ve a "attachments" al final de la página del problema en Kattis. El uso de la herramienta es opcional. Observad que el programa del corrector oficial de Kattis es diferente de la herramienta de testing.

Para usar la herramienta, crea un fichero de entrada, como ahora "sample1.in", que debe empezar con un número N seguido de una línea con N números especificando la permutación, y otra línea con N bits (0 o 1) especificando los estados iniciales de los pájaros. Por ejemplo:

```
6
1 2 0 4 3 5
1 1 0 0 1 0
```

Para programas en Python, como ahora `solution.py` (normalmente ejecutado como `pypy3 solution.py`):

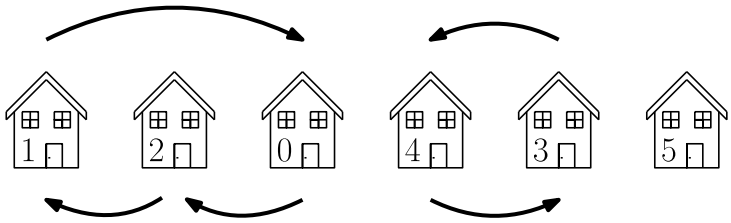
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

Para programas en C++, primero hay que compilar (por ejemplo con mediante `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`) y entonces ejecutar:

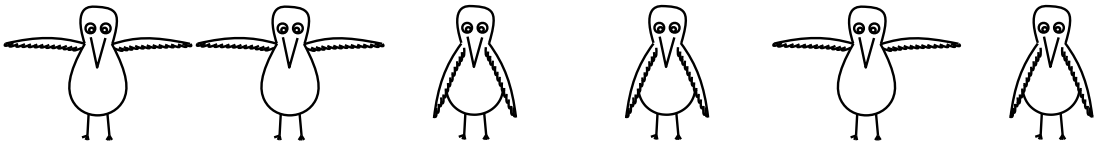
```
python3 testing_tool.py ./solution.out < sample1.in
```

Ejemplo

En el ejemplo, se proporciona la siguiente permutación de personas en las casas:

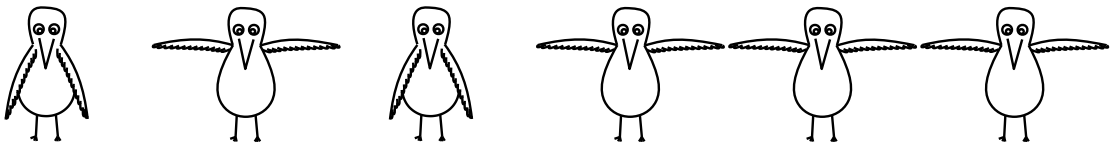


La primera línea del programa de ejemplo se ejecuta (con $w = 0$), e imprime $W = 2$, implicando que Detje pasará por la calle dos veces (y el programa se ejecutará dos veces) Antes del primer paseo, los pájaros en los jardines tienen la siguiente apariencia:



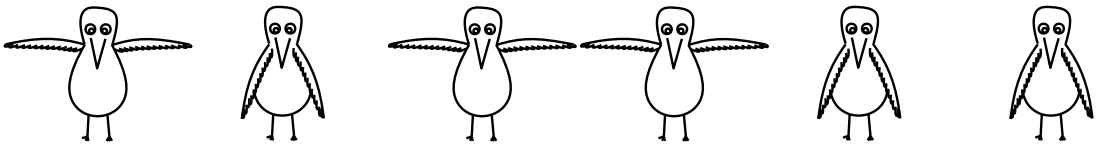
Entonces el programa se ejecuta con $w = 1$, indicando el primer paseo de Detje. Detje pasa por los pájaros uno por uno, empezando por la izquierda y posiblemente cambiando su estado. El programa de ejemplo ha de imprimir el estado del i-ésimo pájaro antes de ver el $(i + 1)$-ésimo pájaro.

Una vez Detje ha llegado a la escuela, los estados de los pájaros son:



En la última ejecución del programa (con $w = 2$), Detje vuelve a casa desde la escuela. Recuerda que en este caso, pasa por los pájaros de derecha a izquierda, ¡procesándolos en orden inverso! Esto significa que debe determinar el estado del pájaro i-ésimo antes de ver el pájaro $(i - 1)$-ésimo.

Tras acabar su paseo, los pájaros tienen la siguiente apariencia:



Efectivamente, esta es la configuración correcta. Por ejemplo, la estatua de pájaro 3 (la cuarta desde la izquierda) está abierta (ahora $b_3 = 1$), lo que es correcto puesto que la persona 4 se

mudará allí ($a_3 = 4$) y originalmente tenía un estado de pájaro abierto (originalmente $b_4 = 1$).

Salida del corrector	Tu salida
0 6	
1 2 0 4 3 5	
	2

Salida del corrector	Tu salida
1 6	
1 2 0 4 3 5	
1	
	0
1	
	1
0	
	0
0	
	1
1	
	1
0	
	1

Salida del corrector	Tu salida
2 6	
1 2 0 4 3 5	
1	
	0
1	
	0
1	
	1
0	
	1
1	
	0
0	
	1