Gondola

La Gondola Mao-Kong es una famosa atracción en Taipei. La góndola consiste en un carril circular, una estación, y n vagones numerados consecutivamente de 1 a n a lo largo del carril en una dirección fija. Después de que el vagón i pasa la estación, el siguiente vagón a pasar será el vagón $i+1$ si $i < n$, o el vagón 1 si $i = n$.

Los vagones se pueden romper. Afortunadamente tenemos un suministro infinito de vagones de repuesto, el cual está numerado $n+1$, $n+2$, y así sucesivamente. Cuando un vagón se rompe, lo reemplazamos (en la misma posición en el carril) con el primer vagón de repuesto disponible, es decir, el vagón con el número menor. Por ejemplo, si hay cinco vagones y el vagón uno 1 se rompe, entonces lo reemplazaremos con el vagón 6.

A ti te gusta pararte en la estación y ver los vagones pasar. Una *secuencia de vagones válida* es una secuencia de n números de vagones que pasan por la estación. Es posible que uno o más vagones se hayan roto (y fueron reemplazados) antes de que tú hayas llegado, pero ningún vagón se rompe mientras estás viendo.

Notar que la misma configuración de vagones en el riel puede dar múltiples secuencias de vagones válidas, dependiendo de cual vagón pasa primero cuando llegas a la estación. Por ejemplo, si ningún vagón se ha roto, entonces ambas secuencias (2,3,4,5,1) y (4,5,1,2,3) son posibles secuencias de vagones, pero (4,3,2,5,1) no lo es (porque los vagones aparecen en el orden equivocado).

Si el vagón 1 se rompe, entonces podemos observar la secuencia de vagones válida (4,5,6,2,3). Si el vagón cuatro se rompe posteriormente, lo reemplazamos con el vagón 7 y podremos observar la secuencia de vagones válida (6,2,3,7,5). Si el vagón 7 se rompe después, lo reemplazamos con el vagón 8 y veremos la secuencia de vagones válida (3,8,5,6,2).

Vagon roto	Nuevo vagón	Posible secuencia de vagones válida
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

Una *secuencia de reemplazo* es una secuencia consistente en el número de los vagones que se han roto, en el orden en que se rompieron. En el ejemplo anterior, la *secuencia de reemplazo* es (1,4,7). Una *secuencia de reemplazo* r *produce* una secuencia de vagones válida g si, después de que los vagones se rompen, de acuerdo a la *secuencia de reemplazo* r, la secuencia de vagones g puede ser observada.

Chequeo de secuencias de vagones

En las primeras tres subtareas debes verificar si una secuencia de entrada es una secuencia de vagones válida. Mirar las tablas de más abajo para tener ejemplos de secuencias que son y que no son

secuencias de vagones válidas. Debes implementar la función `valid`.

- `valid(n, inputSeq)`
  - `n`: Tamaño de la secuencia de entrada
  - `inputSeq`: arreglo de tamaño n; `inputSeq[i]` es el elemento i de la secuencia de entrada, para $0 \leq i \leq n-1$.
  - La función debe retornar 1 si la secuencia de entrada es una secuencia de vagones válida, 0 de otro modo.

Subtareas 1, 2, 3

Subtarea	Puntos	n	`inputSeq`
1	5	$n \leq 100$	Tiene cada número de 1 a n exactamente una vez
2	5	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq n$
3	10	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 250,000$

Ejemplos

Subtarea	`inputSeq`	Valor de retorno	Nota
1	(1, 2, 3, 4, 5, 6, 7)	1	
1	(3, 4, 5, 6, 1, 2)	1	
1	(1, 5, 3, 4, 2, 7, 6)	0	1 no puede aparecer justo antes de 5
1	(4, 3, 2, 1)	0	4 no puede aparecer justo antes de 3
2	(1, 2, 3, 4, 5, 6, 5)	0	Dos vagones numerados 5
3	(2, 3, 4, 9, 6, 7, 1)	1	Secuencia de reemplazo (5, 8)
3	(10, 4, 3, 11, 12)	0	4 4 no puede aparecer justo antes que 3

Secuencia de reemplazo

En las siguientes tres subtareas debes construir una posible secuencia de reemplazo que produce una secuencia de vagones dada. Cualquier secuencia de reemplazo válida será aceptada. Debes implementar una función `replacement`.

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n` es el tamaño de la secuencia de vagones.
  - `gondolaSeq`: Arreglo de tamaño n; `gondolaSeq` es una secuencia de vagones válida, y `gondolaSeq[i]` es el elemento i de la secuencia, para $0 \leq i \leq n-1$.
  - La función deberá retornar l, la longitud de la secuencia de reemplazo.

- `replacementSeq`: Es un arreglo suficientemente largo para almacenar la secuencia de reemplazamiento; usted debe almacenar el elemento i de su secuencia de reemplazo en `replacementSeq[i]`, para $0 \leq i \leq l-1$.

Subtareas 4, 5, 6

Subtarea	Puntos	n	`gondolaSeq`
4	5	$n \leq 100$	$1 \leq$ `gondolaSeq[i]` $\leq n+1$
5	10	$n \leq 1,000$	$1 \leq$ `gondolaSeq[i]` $\leq 5,000$
6	20	$n \leq 100,000$	$1 \leq$ `gondolaSeq[i]` $\leq 250,000$

Ejemplos

Subtarea	`gondolaSeq`	Valor de retorno	`replacementSeq`
4	(3, 1, 4)	1	(2)
4	(5, 1, 2, 3, 4)	0	( )
5	(2, 3, 4, 9, 6, 7, 1)	2	(5, 8)

Contar las secuencias de reemplazamientos

En las siguientes cuatro subtareas debes contar el número de posibles secuencias de reemplazo que producen una secuencia dada (la cual puede o no puede ser una secuencia de vagones valida), la respuesta debe darse módulo **1,000,000,009**. Debes implementar la función `countReplacement`.

- `countReplacement(n, inputSeq)`
  - `n`: Tamaño de la secuencia de entrada
  - `inputSeq`: Arreglo de tamaño n; `inputSeq[i]` es el elemento i de la secuencia de entrada, para $0 \leq i \leq n-1$.
- Si la secuencia de entrada es una secuencia de vagones válida, entonces debe contar el número de secuencias de reemplazo que producen dicha secuencia de vagones (el cual puede ser extremadamente largo), *y retornar este número módulo **1,000,000,009***. Si la secuencia de entrada no es una secuencia de vagones válida, la función deberá retornar 0. Si la secuencia de entrada es una secuencia de vagones válida, pero ningún vagón se rompió, la función deberá retornar 1.

Subtareas 7, 8, 9, 10

Subtarea	Puntos	n	`inputSeq`
7	5	$4 \leq n \leq 50$	$1 \leq$ `inputSeq[i]` $\leq n+3$
8	15	$4 \leq n \leq 50$	$1 \leq$ `inputSeq[i]` ≤ 100, y al menos $n-3$ de los vagones iniciales $1, \ldots, n$ no se rompieron.

Subtarea	Puntos	n	inputSeq
9	15	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 250,000$
10	10	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 1,000,000,000$

Ejemplos

Subtarea	inputSeq	Valor de retorno	Secuencia de reemplazo
7	(1, 2, 7, 6)	2	(3, 4, 5) or (4, 5, 3)
8	(2, 3, 4, 12, 6, 7, 1)	1	(5, 8, 9, 10, 11)
9	(4, 7, 4, 7)	0	inputSeq no es una secuencia de reemplazo
10	(3, 4)	2	(1, 2) or (2, 1)

Detalles de implementación

Debes enviar exactamente un archivo, llamado `gondola.c`, `gondola.cpp` or `gondola.pas`. Este archivo deberá implementar las funciones descritas, usando los siguientes prototipos. Usted también necesita incluir un archivo de cabecera `gondola.h` para implementaciones en C/C++.

Programas en C/C++

```
int valid(int n, int inputSeq[]);
int replacement(int n, int gondolaSeq[], int replacementSeq[]);
int countReplacement(int n, int inputSeq[]);
```

Programas en Pascal

```
function valid(n: longint; inputSeq: array of longint): integer;
function replacement(n: longint; gondolaSeq: array of longint;
var replacementSeq: array of longint): longint;
function countReplacement(n: longint; inputSeq: array of longint):
longint;
```

Grader de ejemplo

El *grader* de ejemplo lee la entrada en el siguiente formato:

- line 1: `T`, el número de subtarea que tu programa debe resolver ($1 \leq T \leq 10$).
- line 2: `n`, la longitud de la secuencia de entrada .
- line 3: Si `T` es 4, 5, or 6, esta línea contiene `gondolaSeq[0]`, ..., `gondolaSeq[n-1]`. de otro modo contiene `inputSeq[0]`, ..., `inputSeq[n-1]`.