



## Gondola (góndola)

La Gondola de Mao-Kong es una famosa atracción en Taipei. El sistema de Gondola consiste en un riel circular, una estación, y  $n$  góndolas numeradas en forma consecutiva desde la 1 a la  $n$  circulando por el riel en una única dirección fija. Después que la góndola  $i$  pasa por la estación, la próxima góndola en pasar la estación será la góndola  $i + 1$  si  $i < n$ , o la góndola 1 si  $i = n$ .

Las góndolas pueden dejar de funcionar. Por suerte tenemos un número infinito de góndolas de repuesto, las cuales están numeradas como  $n + 1$ ,  $n + 2$ , y así sucesivamente. Cuando una góndola se descompone se reemplaza (en la misma posición en el riel) con la primer góndola de repuesto disponible, es decir, la que tiene el número más bajo. Por ejemplo, si hay cinco góndolas y la góndola 1 se rompe, entonces la reemplazaremos con la góndola 6.

A ti te gusta estar en la estación y ver pasar las góndolas. Una *secuencia de góndolas* es una serie de  $n$  números de góndolas que pasan por la estación. Es posible que una o más góndolas se rompan (y fueron sustituidos) antes de que llegaras, pero ninguna de las góndolas se descomponen, mientras tu estás viendo.

Tenga en cuenta que la misma configuración de góndolas en el riel puede dar múltiples secuencias de góndolas, dependiendo de qué góndola pasa primero cuando llegas a la estación. Por ejemplo, si ninguna de las góndolas se ha roto entonces (2, 3, 4, 5, 1) y (4, 5, 1, 2, 3) son posibles secuencias de góndola, pero (4, 3, 2, 5, 1) no es (porque las góndolas aparecen en el orden equivocado).

Si la góndola 1 se rompe, entonces ahora podríamos observar la secuencia de góndolas (4, 5, 6, 2, 3). Si la góndola 4 se rompe luego, la reemplazamos con la góndola 7 y nosotros podríamos observar la secuencia de góndolas (6, 2, 3, 7, 5). Si la góndola 7 se descompone después de esto, se la reemplaza con la góndola 8 y podemos ahora observar la secuencia de góndolas (3, 8, 5, 6, 2).

góndola rota	góndola nueva	posible secuencia de góndolas
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 que consta de los números de las góndolas que se han roto, en el orden en que se descomponen. En el ejemplo anterior la secuencia de reemplazo es (1, 4, 7). Una secuencia de reemplazo  $r$  produce una secuencia de góndolas  $g$  si, después que las góndolas se rompen de acuerdo a la secuencia  $r$ , se puede observar la secuencia de góndolas  $g$ .

## Comprobación de la secuencia de góndolas

En las primeras tres subtarefas debes comprobar si una secuencia de entrada es una secuencia de góndolas. Consulte la tabla siguiente para ver ejemplos de secuencias que son y no son secuencias de góndola. Debes implementar la función `valid`.

- `valid(n, inputSeq)`
  - $n$ : es el largo de la secuencia de entrada.
  - `inputSeq`: array de tamaño  $n$ ; `inputSeq[i]` es el elemento  $i$  de la secuencia de entrada, para  $0 \leq i \leq n - 1$ .
  - La función debe devolver 1 si la secuencia de entrada es una secuencia de góndolas, o 0 en caso contrario.

### Subtareas 1, 2, 3

subtarea	puntos	$n$	<code>inputSeq</code>
1	5	$n \leq 100$	tiene cada número de 1 a $n$ exactamente una vez
2	5	$n \leq 100.000$	$1 \leq \text{inputSeq}[i] \leq n$
3	10	$n \leq 100.000$	$1 \leq \text{inputSeq}[i] \leq 250.000$

### Ejemplos

subtarea	<code>inputSeq</code>	valor retornado	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 del 5
1	(4, 3, 2, 1)	0	4 no puede aparecer justo antes del 3
2	(1, 2, 3, 4, 5, 6, 5)	0	dos góndolas numeradas 5
3	(2, 3, 4, 9, 6, 7, 1)	1	secuencia de reemplazo (5, 8)
3	(10, 4, 3, 11, 12)	0	4 no puede aparecer justo antes del 3

## Secuencia de Reemplazo

En las próximas tres subtareas debes construir una posible secuencia de reemplazo que genera una secuencia de góndolas dada. Se aceptará cualquier secuencia de sustitución. Debes implementar una función `replacement`.

- `replacement(n, gondolaSeq, replacementSeq)`
  - $n$  es el largo de la secuencia de góndolas.
  - `gondolaSeq`: array de tamaño  $n$ ; `gondolaSeq` es garantizado que sea una secuencia de góndolas, y `gondolaSeq[i]` es el elemento  $i$  de la secuencia, para  $0 \leq i \leq n - 1$ .
  - La función debe retornar  $l$ , el largo de la secuencia de reemplazo.

- `replacementSeq`: vector que es lo suficientemente grande para almacenar la secuencia de reemplazo; usted debe devolver su secuencia mediante la colocación de elementos  $i$  de su secuencia de reemplazo en `replacementSeq[i]`,  $0 \leq i \leq l - 1$ .

## Subtareas 4, 5, 6

subtarea	puntos	$n$	<code>gondolaSeq</code>
4	5	$n \leq 100$	$1 \leq \text{gondolaSeq}[i] \leq n + 1$
5	10	$n \leq 1.000$	$1 \leq \text{gondolaSeq}[i] \leq 5.000$
6	20	$n \leq 100.000$	$1 \leq \text{gondolaSeq}[i] \leq 250.000$

## Ejemplos

subtarea	<code>gondolaSeq</code>	valor de retorno	<code>replacementSeq</code>
4	(3, 1, 4)	1	(2)
4	(5, 1, 2, 3, 4)	0	( )
5	(2, 3, 4, 9, 6, 7, 1)	2	(5, 8)

## Contando Secuencias de Reemplazo

En las próximas cuatro subtareas debes contar el número de posibles secuencias de reemplazo que producen una secuencia dada (que puede o no ser una secuencia de góndolas), modulo **1.000.000.009**. Debes implementar la función `countReplacement`.

- `countReplacement(n, inputSeq)`
  - $n$ : es el largo de la secuencia de entrada.
  - `inputSeq`: array 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 góndolas, entonces contar el número de secuencias de reemplazo que producen esta secuencia de góndolas (que podría ser extremadamente grande), y *retornar este número modulo 1.000.000.009*. Si la secuencia de entrada no es una secuencia de góndolas, la función debe devolver 0. Si la secuencia de entrada es una secuencia de góndolas, pero no hay góndolas rotas, la función debe devolver 1.

## Subtareas 7, 8, 9, 10

subtarea	puntos	$n$	<code>inputSeq</code>
7	5	$4 \leq n \leq 50$	$1 \leq \text{inputSeq}[i] \leq n + 3$
8	15	$4 \leq n \leq 50$	$1 \leq \text{inputSeq}[i] \leq 100$ , y al menos $n - 3$ de las góndolas iniciales $1, \dots, n$ no se rompen.

subtarea	puntos	$n$	inputSeq
9	15	$n \leq 100.000$	$1 \leq \text{inputSeq}[i] \leq 250.000$
10	10	$n \leq 100.000$	$1 \leq \text{inputSeq}[i] \leq 1.000.000.000$

## Ejemplos

subtarea	inputSeq	valor de retorno	secuencia de reemplazo
7	(1, 2, 7, 6)	2	(3, 4, 5) o (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 góndolas
10	(3, 4)	2	(1, 2) o (2, 1)

## Detalles de la implementación

Debes enviar exactamente un archivo, llamado `gondola.c`, `gondola.cpp` o `gondola.pas`. Este archivo debiera implementar el subprograma descrito más arriba, utilizando los siguientes encabezamientos. Necesitas incluir también un archivo de cabecera `gondola.h` para la implementación en C/C++.

### Programación en C/C++

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

### Programación 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;
```

### Sample grader

El sample grader (programa evaluador para la prueba local) lee su entrada con el formato siguiente:

- línea 1:  $T$ , el número de subtarea que tu programa intenta resolver ( $1 \leq T \leq 10$ ).
- línea 2:  $n$ , el largo de la secuencia de entrada.
- línea 3: Si  $T$  es 4, 5, o 6, esta línea contiene `gondolaSeq[0], ..., gondolaSeq[n-1]`. De lo contrario esta línea contiene `inputSeq[0], ..., inputSeq[n-1]`.