# Gondola (Vagones)

El teleférico Mao-Kong es una atracción famosa en Taipei. El sistema del teleférico consiste en un riel circular, una sola estación, y $n$ vagones numerados consecutivamente desde 1 hasta $n$ que corren a lo largo del riel en una dirección fija. Después que el vagón $i$ pasa por la estación, el siguiente vagón en pasar por la estación será el vagón $i+1$ si $i < n$, o el vagón 1 si $i = n$.

Los vagones se pueden averiar. Por fortuna, tenemos una cantidad infinita de vagones de respuesto, que están numerados $n+1$, $n+2$, y así sucesivamente. Cuando un vagón se avería, lo reemplazamos (en la misma posición en el riel) con el primer vagón de repuesto disponible, esto es, el que tenga el número más bajo. Por ejemplo, si hay cinco vagones y el vagón 1 se avería, lo reemplazaremos con el vagón 6.

A tí te gusta pararte en la estación y ver los vagones pasar. Una *secuencia de vagones válida* es una secuencia de $n$ números correspondientes a los vagones que pasan por la estación. Es posible que uno o más vagones se hayan averiado (y ya hayan sido reemplazados) antes de que llegaras a la estación, pero ninguno de los vagones se averiará mientras estés viendo los vagones en la estación.

Toma en cuenta que la misma configuración de vagones en el riel puede producir múltiples secuencias de vagones válidas, dependiendo de qué vagón pasa primero cuando llegas a la estación. Por ejemplo, si ninguno de los vagones se ha averiado, entonces tanto (2, 3, 4, 5, 1) como (4, 5, 1, 2, 3) son posibles secuencias de vagones válidas, pero (4, 3, 2, 5, 1) no lo es (porque los vagones aparecen en un orden incorrecto).

Si el vagón 1 se avería, entonces podríamos observar la secuencia de vagones (4, 5, 6, 2, 3). Si entonces se averiara el vagón 4, lo reemplazaríamos con el vagón 7 y podríamos observar la secuencia (6, 2, 3, 7, 5). Si después se avería el vagón 7, podríamos reemplazarlo con el vagón 8 y una secuencia posible que podríamos observar sería (3, 8, 5, 6, 2).

| vagón averiado | nuevo vagón | posible secuencia de vagones |
|---|---|---|
| 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 consiste en los números de los vagones que se han averiado, en el orden en el que se averiaron. En el ejemplo previo, la secuencia de reemplazo sería (1, 4, 7). Una secuencia de reemplazo $r$ *produce* una secuencia válida de vagones $g$ si, después de que los vagones se averían de acuerdo a la secuencia de reemplazo $r$, la secuencia de vagones $g$ puede ser observada.

## Verificando secuencias de vagones

En las primeros tres subtareas debes verificar si una secuencia de entrada es una secuencia de vagones válida. Observa la tabla de abajo para encontrar ejemplos de secuencias que no son

secuencias de vagones. Debes implementar una función llamada `valid`.

- `valid(n, inputSeq)`
  - `n`: la longitud de la secuencia de entrada.
  - `inputSeq`: un arreglo de longitud $n$; `inputSeq[i]` es el $i$-ésimo elemento 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 vagón válida, o 0 en el caso contrario.

### 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 devuelto | 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 | hay dos vagones con el número 5 |
| 3 | (2, 3, 4, 9, 6, 7, 1) | 1 | la secuencia de reemplazo es (5, 8) |
| 3 | (10, 4, 3, 11, 12) | 0 | 4 no puede aparecer justo antes de 3 |

# Secuencia de reemplazo

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

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n` es la longitud de la secuencia de vagones.
  - `gondolaSeq`: un arreglo de longitud $n$; `gondolaSeq` está garantizado que es una secuencia de vagón válida, y `gondolaSeq[i]` es el $i$-ésimo elemento de la secuencia, para $0 \leq i \leq n-1$.
  - La función debe devolver $l$, la longitud de la secuencia de reemplazo.
  - `replacementSeq`: un arreglo lo suficientemente largo para almacenar la secuencia de

reemplazo; debes devolver tu secuencia colocando el $i$-ésimo elemento de tu 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 devuelto | `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) |

# Contando secuencias de reemplazo

En las siguientes cuatro subtareas debes contar el número de posibles secuencias de reemplazo que producen la secuencia de vagones dada inicialmente (que puede o no puede ser una secuencia de vagones válida), modulo **1,000,000,009**. Tienes que implementar la función `countReplacement`.

- `countReplacement(n, inputSeq)`
  - `n`: la longitud de la secuencia de entrada.
  - `inputSeq`: arreglo de tamaño $n$; `inputSeq[i]` es el elemento $i$-ésimo de la secuencia de entrada, donde $0 \leq i \leq n-1$.
  - Si la secuencia de entrada es una secuencia de vagones válida, entonces calcula el número de secuencias de reemplazo que producen la secuencia de entrada *y devuelve este número modulo **1,000,000,009***. Si la secuencia de entrada no es una secuencia de vagones válida, la función debe devolver 0. Si la secuencia de entrada es una secuencia de vagones válida pero ningun vagón se averió, entonces la función debe devolver 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]` $\leq 100$, y al menos $n-3$ de los vagones iniciales $1, \ldots, n$ no se averiaron. |
| 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 devuelto | 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 vagones válida |
| 10 | (3, 4) | 2 | (1, 2) or (2, 1) |

# Detalles de implementación

Tienes que mandar exactamente un archivo llamado `gondola.c`, `gondola.cpp` o `gondola.pas`. Este archivo debe tener implementadas las 3 funciones descritas anteriormente (incluso si sólo planeas resolver una de las subtareas), usando los siguientes prototipos. También debes incluir el archivo `gondola.h` para las 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;
```

### Evaluador de ejemplo

El evaluador de ejemplo lee la entrada en el siguiente formato:

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