Gondola

La Mao-Kong Gondola è una famosa attrazione di Taipei. Il sistema è composto da un binario circolare, da un'unica stazione e da n gondole numerate da 1 a n, che circolano lungo il binario in una direzione prefissata. Nella situazione iniziale, dopo che la gondola i è passata per la stazione, la successiva a passare sarà la gondola $i + 1$ se $i < n$, altrimenti la gondola 1 se $i = n$.

Le gondole possono rompersi. Fortunatamente, abbiamo infinite gondole di ricambio, numerate $n + 1$, $n + 2$ e così via. Quando una gondola si rompe, la rimpiazziamo (nella stessa posizione sul binario) con la prima gondola di rimpiazzo disponibile (cioè quella avente numero minore). Per esempio, se ci sono cinque gondole e la gondola 1 si rompe, la rimpiazzeremo con la gondola 6.

Ti piace stare alla stazione e guardare le gondole passare. Una *successione di gondole* è una successione di n numeri di gondole che passano per la stazione. È possibile che una o più gondole si siano rotte (e siano state rimpiazzate) prima del tuo arrivo, ma nessuna delle gondole si rompe mentre stai guardando.

Nota che la stessa configurazione di gondole sul binario può creare diverse successioni di gondole, a seconda di quale gondola passa per prima dopo il tuo arrivo alla stazione. Per esempio, se nessuna gondola si è rotta, allora sia (2, 3, 4, 5, 1) che (4, 5, 1, 2, 3) sono successioni di gondole possibili, ma (4, 3, 2, 5, 1) non lo è (perché le gondole compaiono nell'ordine sbagliato).

Se la gondola 1 si rompe, allora possiamo osservare la successione di gondole (4, 5, 6, 2, 3). Se successivamente si rompe la gondola 4, la rimpiazziamo con la gondola 7 e possiamo osservare la successione di gondole (6, 2, 3, 7, 5). Se dopo ciò si rompe la gondola 7, la rimpiazziamo con la gondola 8 e possiamo osservare la successione di gondole (3, 8, 5, 6, 2).

gondola rotta	gondola di rimpiazzo	possibile successione di gondole
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

Una *successione di rimpiazzi* è una successione composta da numeri di gondole che si sono rotte, nell'ordine in cui si sono rotte. Nell'esempio precedente, la successione di rimpiazzi è (1, 4, 7). Una successione di rimpiazzi r *produce* una successione di gondole g se, dopo che le gondole si sono rotte secondo la successione di rimpiazzi r, la successione di gondole g può essere osservata.

Controllare una successione di gondole

Nei primi tre subtask, devi controllare se una successione data in input è una successione di gondole. La tabella di seguito contiene esempi di successioni che sono o non sono successioni di gondole. Devi implementare una funzione `valid`.

- `valid(n, inputSeq)`
  - `n`: lunghezza della successione in input.
  - `inputSeq`: array di lunghezza n; `inputSeq[i]` è l'elemento i della successione in input, per $0 \leq i \leq n-1$.
  - La funzione deve restituire 1 se la successione di input è una successione di gondole, 0 altrimenti.

Subtask 1, 2, 3

subtask	punti	n	`inputSeq`
1	5	$n \leq 100$	ogni numero da 1 a n compare esattamente una volta
2	5	$n \leq 100\,000$	$1 \leq$ `inputSeq[i]` $\leq n$
3	10	$n \leq 100\,000$	$1 \leq$ `inputSeq[i]` $\leq 250\,000$

Esempi

subtask	`inputSeq`	valore restituito	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 non può apparire immediatamente prima di 5
1	(4, 3, 2, 1)	0	4 non può apparire immediatamente prima di 3
2	(1, 2, 3, 4, 5, 6, 5)	0	due gondole numerate con 5
3	(2, 3, 4, 9, 6, 7, 1)	1	la successione di rimpiazzo è (5, 8)
3	(10, 4, 3, 11, 12)	0	4 non può apparire immediatamente prima di 3

Successione di rimpiazzo

Nei successivi tre subtask, devi costruire una possibile successione di rimpiazzi che produce una data successione di gondole. Una qualsiasi di queste successioni di rimpiazzi sarà accettata. Devi implementare una funzione `replacement`.

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n`: lunghezza della successione di gondole.
  - `gondolaSeq`: array di lunghezza n; è garantito che `gondolaSeq` è una successione di gondole, e `gondolaSeq[i]` è l'elemento i della successione, per $0 \leq i \leq n-1$.
  - La funzione deve restituire l, la lunghezza della successione di rimpiazzi.
  - `replacementSeq`: array abbastanza largo da contenere la successione di rimpiazzi; devi restituire la tua successione piazzando l'elemento i della tua successione di rimpiazzi in `replacementSeq[i]`, per $0 \leq i \leq l-1$.

Subtask 4, 5, 6

subtask	punti	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]` $\leq 250\,000$

Esempi

subtask	gondolaSeq	valore restituito	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)

Contare successioni di rimpiazzi

Nei successivi quattro subtask, devi contare il numero di successioni di rimpiazzi che possono aver prodotto una data successione (che non necessariamente è una successione di gondole), modulo **1 000 000 009**. Devi implementare una funzione `countReplacement`.

- `countReplacement(n, inputSeq)`
  - `n`: lunghezza della successione in input.
  - `inputSeq`: array di lunghezza n; `inputSeq[i]` è l'elemento i della successione di input, per $0 \leq i \leq n-1$.
  - Se la successione in input è una successione di gondole, allora conta il numero di successioni di rimpiazzi che producono la successione di gondole (tale numero può essere estremamente grande), *e restituisci questo numero modulo **1 000 000 009***. Se la successione in input non è una successione di gondole, la funzione deve restituire 0. Se la successione in input è una successione di gondole, ma nessuna gondola si è rotta, la funzione deve restituire 1.

Subtask 7, 8, 9, 10

subtask	punti	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, e almeno $n-3$ delle gondole di partenza in $1, \ldots, n$ non si sono rotte.
9	15	$n \leq 100\,000$	$1 \leq$ `inputSeq[i]` $\leq 250\,000$
10	10	$n \leq 100\,000$	$1 \leq$ `inputSeq[i]` $\leq 1\,000\,000\,000$

Esempi

subtask	`inputSeq`	valore restituito	successione di rimpiazzi
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` non è una successione di gondole
10	(3, 4)	2	(1, 2) o (2, 1)

Dettagli di implementazione

Devi sottoporre esattamente un file, chiamato `gondola.c`, `gondola.cpp` o `gondola.pas`. Questo file deve implementare tutte e tre le procedure descritte sopra (anche se prevedi di risolvere solo parte dei subtask), usando l'intestazione seguente. In C/C++, devi anche includere il file header `gondola.h`.

Linguaggio C/C++

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

Linguaggio 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 di esempio

Il grader di esempio legge l'input secondo il formato seguente:

- riga 1: `T`, il numero di subtask che il tuo programma vuole risolvere ($1 \leq T \leq 10$).
- riga 2: `n`, la lunghezza della successione in input.
- riga 3: se `T` è 4, 5, o 6, questa riga conterrà `gondolaSeq [0]`, ..., `gondolaSeq [n-1]`. Altrimenti, questa riga conterrà `inputSeq [0]`, ..., `inputSeq [n-1]`.