Gondola

Mao-Kong Gondola is een beroemde attractie in Taipei. Het gondel-systeem bestaat uit een cirkelvormige rail, één station, en n gondels die achtereenvolgens genummerd zijn van 1 t/m n en die in een vaste richting over de rails rijden. Nadat gondel i is gepasseerd, is de volgende gondel die het station passeert $i + 1$ wanneer $i < n$, maar gondel 1 wanneer $i = n$.

Gondels kunnen stuk gaan. Er is een oneindige voorraad reserve gondels, genummerd $n + 1$, $n + 2$, enz. Wanneer een gondel stuk gaat, wordt deze vervangen (op dezelfde plaats in de rij) door de eerst beschikbare reserve gondel, degene met het laagste getal. Wanneer er bijvoorbeeld vijf gondels zijn en gondel 1 gaat stuk, dan wordt deze vervangen door gondel 6.

Je vindt het leuk om op het station de gondels te zien passeren. Een *gondel-rij* is een rij van n getallen die gondels representeren die het station passeren. Het is mogelijk dat een of meer gondels zijn stuk gegaan (en vervangen) voordat je aankwam, maar geen van de gondels gaat stuk terwijl je staat te kijken.

Dezelfde samenstelling van gondels op de rail kan meerdere gondel-rijen opleveren. Als er geen gondels zijn stuk gegaan dan zijn zowel (2,3,4,5,1) als (4,5,1,2,3) mogelijke gondel-rijen; (4,3,2,5,1) is dat niet omdat de gondels in een verkeerde volgorde verschijnen.

Wanneer gondel 1 stuk gaat, kun je de gondel-rij (4,5,6,2,3) zien passeren. Als vervolgens gondel 4 stuk gaat, dan wordt deze vervangen door gondel 7 en kan bijvoorbeeld gondel-rij (6,2,3,7,5) passeren. Wanneer daarna gondel 7 stuk gaat wordt deze vervangen door gondel 8 en zie je wellicht gondel-rij (3,8,5,6,2) passeren.

gondel die stuk is	nieuwe gondel	mogelijke gondel-rij
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

Een *vervangingsrij* is een een rij die bestaat uit de getallen behorende bij de gondels die zijn stuk gegaan, in de volgorde waarin ze zijn stuk gegaan. In het vorige voorbeeld is de vervangingsrij (1,4,7). Een vervangingsrij r *produceert* een gondel-rij g wanneer, nadat gondels zijn stuk gegaan volgens de vervangingsrij r, de gondel-rij g kan passeren.

Gondel-rij check

In de eerste drie subtasks moet je checken of een input rij een gondel-rij is. Kijk naar de tabel hieronder die bestaat uit rijen die wel of niet gondel-rijen zijn. Je moet een functie `valid` implementeren:

- `valid(n, inputSeq)`
  - `n`: de lengte van de input rij
  - `inputSeq`: array van lengte n; `inputSeq[i]` is het i-de element van de input rij, met $0 \leq i \leq n-1$.
  - De functie moet als resultaat 1 geven wanneer de invoerrij een gondel-rij is, en anders 0.

Subtasks 1, 2, 3

subtask	punten	n	`inputSeq`
1	5	$n \leq 100$	bevat elk getal van 1 tot en met n precies een keer
2	5	$n \leq 100.000$	$1 \leq$ `inputSeq[i]` $\leq n$
3	10	$n \leq 100.000$	$1 \leq$ `inputSeq[i]` ≤ 250.000

Voorbeelden

subtask	`inputSeq`	resultaat	opmerking
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 mag niet precies voor 5 verschijnen
1	(4, 3, 2, 1)	0	4 mag niet precies voor 3 verschijnen
2	(1, 2, 3, 4, 5, 6, 5)	0	twee gondels hebben nummer 5
3	(2, 3, 4, 9, 6, 7, 1)	1	vervangingsrij (5, 8)
3	(10, 4, 3, 11, 12)	0	4 mag niet precies voor de 3 verschijnen

Vervangingsrij

Bij de volgende drie subtasks moet je een mogelijke vervangingsrij construeren die een gegeven gondel-rij produceert. Iedere mogelijke vervangingsrij wordt geaccepteerd. Implementeer de functie `replacement`:

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n` is de lengte van de gondel-rij.
  - `gondolaSeq`: array van lengte n; `gondolaSeq` is gegarandeerd een gondel-rij, en `gondolaSeq[i]` is element i van de rij, waarbij $0 \leq i \leq n-1$.
  - De functie moet als resultaat l geven, de lengte van de vervangingsrij.
  - `replacementSeq`: array die voldoende groot is om de vervangingsrij in op te slaan. Sla element i van je vervangingsrij op in `replacementSeq[i]`, met $0 \leq i \leq l-1$.

Subtasks 4, 5, 6

subtask	punten	n	gondolaSeq
4	5	$n \leq 100$	$1 \leq$ gondolaSeq[i] $\leq n+1$
5	10	$n \leq 1.000$	$1 \leq$ gondolaSeq[i] ≤ 5.000
6	20	$n \leq 100.000$	$1 \leq$ gondolaSeq[i] ≤ 250.000

Voorbeelden

subtask	gondolaSeq	resultaat	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)

Tel De Vervangingsrijen

In de volgende vier subtasks moet je het aantal mogelijke vervangingsrijen bepalen die een gegeven rij kunnen produceren (die al dan niet een gondel-rij zijn) modulo **1.000.000.009**. Je moet een functie `countReplacement` implementeren.

- `countReplacement(n, inputSeq)`
  - `n`: de lengte van de input rij.
  - `inputSeq`: array met lengte n; `inputSeq[i]` is element i van de input rij, waarbij $0 \leq i \leq n-1$.
  - Als de invoerrij een gondel-rij is, tel dan het aantal vervangingsrijen die deze gondel-rij produceert (dat kan extreem groot zijn), *en return dit getal modulo* ***1.000.000.009***. Wanneer de invoer rij geen gondel-rij is moet de functie 0 als resultaat geven. Wanneer de invoer rij een gondel-rij is, maar geen van de gondels ging stuk, dan moet de functie 1 als resultaat geven.

Subtasks 7, 8, 9, 10

subtask	punten	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, en tenminste $n-3$ van de initiele gondels $1, \ldots, n$ gingen niet stuk.
9	15	$n \leq 100.000$	$1 \leq$ inputSeq[i] ≤ 250.000
10	10	$n \leq 100.000$	$1 \leq$ inputSeq[i] $\leq 1.000.000.000$

Voorbeelden

subtask	inputSeq	resultaat	vervangingsrij
7	(1, 2, 7, 6)	2	(3, 4, 5) of (4, 5, 3)
8	(2, 3, 4, 12, 6, 7, 1)	1	(5, 8, 9, 10, 11)
9	(4, 7, 4, 7)	0	`inputSeq` is geen gondel-rij
10	(3, 4)	2	(1, 2) of (2, 1)

Implementatie details

Je moet precies een bestand submitten, met de naam `gondola.c`, `gondola.cpp` of `gondola.pas`. Dit bestand moeten al de drie subprogramma's implementeren zoals boven beschreven (ook wanneer je alleen maar van plan een paar subtasks op te lossen); gebruik de volgende signatures. Bij de C/C++ implementatie moet je ook de header file `gondola.h` includen.

C/C++ programma's

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

Pascal programs

```
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

De sample grader leest de input in het volgende format:

- line 1: `T`, het nummer van de subtask number dat je programma wil oplossen ($\mathbf{1 \leq T \leq 10}$).
- line 2: `n`, de lengte van de invoerrij.
- line 3: Wanneer `T` 4, 5, of 6 is, dan bevat deze regel `gondolaSeq[0]`, ..., `gondolaSeq[n-1]`. Anders bevat deze regel `inputSeq[0]`, ..., `inputSeq[n-1]`.