Gondola

De Mao-Kong Gondola (kabelbaan) is een beroemde attractie in Taipei. De kabelbaan bestaat uit een circulaire kabel, één station, en n gondels (cabines) die opeenvolgend van 1 tot n genummerd zijn en in een vaste richting de kabel volgen. Nadat gondel i het station voorbij is, zal de volgende gondel die het station passeert gondel $i + 1$ zijn als $i < n$, of gondel 1 als $i = n$.

Gondels kunnen kapotgaan. Gelukkig hebben we een oneindige voorraad reservegondels, genummerd $n + 1$, $n + 2$, enzovoort. Als een gondel het begeeft dan vervangen we die (op dezelfde positie op de kabelbaan) met de eerst beschikbare reservegondel, dat is diegene met het laagste nummer. Dus, als er 5 gondels zijn en gondel 1 gaat kapot, dan vervangen we die door gondel 6.

Je staat graag aan het station om naar de passerende gondels te kijken. Een *gondelreeks* is een reeks van n gondelnummers die door het station passeren. Het is mogelijk dat één of meer gondels het al hadden begeven (en waren vervangen) voordat je aankwam, maar er zal nooit een gondel kapotgaan terwijl je kijkt.

Zo kan dezelfde configuratie van gondels aan de kabel verschillende gondelreeksen geven, afhankelijk van welke gondel eerst passeert wanneer je bij het station aankomt. Bijvoorbeeld, als er geen gondels kapot zijn gegaan dan zijn (2, 3, 4, 5, 1) en (4, 5, 1, 2, 3) mogelijke gondelreeksen, maar (4, 3, 2, 5, 1) is niet mogelijk (want de gondels staan niet op volgorde).

Als gondel 1 het begaf, observeren we mogelijk de gondelreeks (4, 5, 6, 2, 3). Als daarna gondel 4 het begeeft, werd die vervangen door gondel 7 en observeren we mogelijk de reeks (6, 2, 3, 7, 5). Als daarna gondel 7 in panne valt, vervangen we die door gondel 8 en observeren we mogelijk de reeks (3, 8, 5, 6, 2).

kapotte gondel	nieuwe gondel	mogelijke gondelreeks
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

Een *vervangingsreeks* is een reeks die bestaat uit de nummers van de gondels die kapot zijn gegaan, in de volgorde waarin ze kapot zijn gegaan. In het vorige voorbeeld is de vervangingsreeks (1, 4, 7). Een vervangingsreeks r *produceert* een gondelreeks g als, nadat de gondels zijn kapotgegaan volgens vervangingsreeks r, de gondelreeks g mogelijk geobserveerd kan worden.

Gondelreeksen controleren

In de eerste drie subtaken moet je controleren of een gegeven reeks een gondelreeks is. In de tabel hieronder staan voorbeelden van reeksen die wel of niet gondelreeksen zijn. Je moet een functie `valid` implementeren.

- `valid(n, inputSeq)`
  - `n`: de lengte van de gegeven reeks.
  - `inputSeq`: array van lengte n; `inputSeq[i]` is element i van de gegeven reeks, voor $0 \leq i \leq n-1$.
  - De functie moet 1 teruggeven als de gegeven reeks een gondelreeks is, en anders 0.

Subtaken 1, 2, 3

subtaak	punten	n	`inputSeq`
1	5	$n \leq 100$	bevat elk getal van 1 tot n exact éénmaal
2	5	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq n$
3	10	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 250,000$

Voorbeelden

subtaak	`inputSeq`	return-waarde	opmerkingen
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 kan niet vlak voor 5 komen
1	(4, 3, 2, 1)	0	4 kan niet vlak voor 3 komen
2	(1, 2, 3, 4, 5, 6, 5)	0	twee gondels met nummer 5
3	(2, 3, 4, 9, 6, 7, 1)	1	gebruikte vervangingsreeks: (5, 8)
3	(10, 4, 3, 11, 12)	0	4 kan niet vlak voor 3 komen

Vervangingsreeksen

In de 3 volgende subtaken moet je een mogelijke vervangingsreeks construëren die een gegeven gondelreeks produceert. Elke vervangingsreeks die dat bereikt zal worden geaccepteerd. Je moet een functie `replacement` implementeren.

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n` is de lengte van de gondelreeks.
  - `gondolaSeq`: array van lengte n; `gondolaSeq` is gegarandeerd een geldige gondelreeks, en `gondolaSeq[i]` is element i van die reeks, voor $0 \leq i \leq n-1$.
  - De functie moet l teruggeven, de lengte van de vervangingsreeks.
  - `replacementSeq`: array die groot genoeg is om de vervangingsreeks te bevatten; je moet jouw reeks teruggeven door element i van jouw vervangingsreeks te schrijven in `replacementSeq[i]`, voor $0 \leq i \leq l-1$.

Subtaken 4, 5, 6

subtaak	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] $\leq 5,000$
6	20	$n \leq 100,000$	$1 \leq$ gondolaSeq[i] $\leq 250,000$

Voorbeelden

subtaak	gondolaSeq	return-waarde	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 vervangingsreeksen

Voor de volgende 4 subtaken moet je het aantal mogelijke vervangingsreeksen tellen die een gegeven reeks kunnen produceren (die wel of niet een geldige gondelreeks kan zijn), modulo **1,000,000,009**. Je moet een functie `countReplacement` implementeren

- `countReplacement(n, inputSeq)`
  - `n`: de lengte van de inputreeks.
  - `inputSeq`: array van lengte n; `inputSeq[i]` is element i van de inputreeks, voor $0 \leq i \leq n-1$.
  - Als de inputreeks een gondelreeks is, dan moet je het aantal vervangingsreeksen tellen die deze gondelreeks produceren (dit kan een extreem groot getal zijn), *en dit getal teruggeven modulo **1,000,000,009***. Als de inputreeks geen gondelreeks is, moet de functie 0 teruggeven. Als de inputreeks een gondelreeks is maar geen enkele gondel is kapotgegaan, moet de functie 1 teruggeven.

Subtaken 7, 8, 9, 10

subtaak	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 minstens $n-3$ van de initiële gondels $1, \ldots, n$ gingen niet kapot.
9	15	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 250,000$
10	10	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 1,000,000,000$

Voorbeelden

subtaak	`inputSeq`	return-waarde	vervangingsreeks(en)
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 gondelreeks
10	(3, 4)	2	(1, 2) of (2, 1)

Implementatiedetails

Je moet exact één bestand indienen, genaamd `gondola.c`, `gondola.cpp` of `gondola.pas`. Dit bestand moet alle drie subroutines implementeren die hierboven beschreven zijn (zelfs als je maar enkele van de subtaken wilt oplossen), volgens de volgende declaraties. In een C/C++ implementatie moet je ook een header-file `gondola.h` "includeren".

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 programma's

```
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;
```

Voorbeeldgrader

De voorbeeldgrader leest input in het volgende formaat:

- lijn 1: `T`, de subtaak die jouw programma tracht op te lossen ($\mathbf{1 \leq T \leq 10}$).
- lijn 2: `n`, de lengte van de inputreeks.
- lijn 3: Als `T` gelijk is aan 4, 5, of 6, dan bevat deze lijn `gondolaSeq[0]`, ..., `gondolaSeq[n-1]`. Anders bevat deze lijn `inputSeq[0]`, ..., `inputSeq[n-1]`.