Seilbahn

Die Mao-Kong Seilbahn ist eine bekannte Sehenswürdigkeit in Taipei. Das Seilbahnsystem besteht aus einem geschlossenen Seil, einer einzelnen Station und n Gondeln, die fortlaufend von 1 bis n nummeriert sind. Des Weiteren werden die Gondeln immer in die selbe Richtung bewegt. Als die Seilbahn erbaut wurde, waren die Gondeln derart angeordnet, dass nach der Gondel i die Gondel $i + 1$ die Station passiert (wenn $i < n$). Nach Gondel n passiert Gondel 1 die Station.

Gondeln können defekt werden. Zum Glück gibt es ausreichend viele Ersatzgondeln, die mit $n + 1$, $n + 2$, ... fortlaufend nummeriert sind.
Wenn eine Gondel defekt wird, wird sie mit der ersten verfügbaren Ersatzgondel - dies ist jene Ersatzgondel mit der kleinsten noch nicht verwendeten Nummer - an der selben Stelle im Seil ersetzt.

Wenn es zum Beispiel 5 Gondeln gibt und Gondel 1 defekt wird, so wird sie durch Gondel 6 ersetzt.

Wenn du in der Station stehst und die vorbeifahrenden Gondeln beobachtest, so ist eine *Gondelsequenz* eine Folge von n Nummern der n vorbeifahrenden Gondeln. Es ist möglich, dass eine oder mehrere Gondeln defekt wurden (und ersetzt wurden) bevor du mit deiner Beobachtung begonnen hast. Es ist garantiert, dass keine der n Gondeln defekt wird, wärend eine *Gondelsequenz* ermittelt wird.

Beachte, dass die selbe Anordnung von Gondeln auf dem Seil verschiedene Gondelsequenzen zur Folge haben, abhängig davon, welche Gondel zu Beginn deiner Beobachtung die Station passiert. Wenn zum Beispiel noch keine Gondel defekt wurde, dann sind sowohl (2, 3, 4, 5, 1) als auch (4, 5, 1, 2, 3) mögliche Gondelsequenzen, während (4, 3, 2, 5, 1) nicht möglich ist, da die Gondeln in der falschen Reihenfolge stehen.

Wenn Gondel 1 defekt wird, dann kann die Gondelsequenz (4, 5, 6, 2, 3) beobachtet werden. Wenn danach auch Gondel 4 defekt wird, so wird sie durch Gondel 7 ersetzt und man kann die Gondelsequenz (6, 2, 3, 7, 5) beobachten. Wenn danach Gondel 7 defekt wird, wird sie durch Gondel 8 ersetzt und man kann die Gondelsequenz (3, 8, 5, 6, 2) beobachten.

Defekte Gondel	Ersatzgondel	Mögliche Gondelsequenz
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

Eine *"Replacement"-Sequenz* (dt. sinngemäss "Defektsequenz") ist eine Folge von Nummern der defekt gewordenen Gondeln, in der Reihenfolge, in der sie defekt wurden.
Im vorigen Beispiel ist die "Replacement"-Sequenz (1, 4, 7). Eine "Replacement"-Sequenz r *erzeugt* eine Gondelsequenz g, wenn die Gondeln gemäss der "Replacement"-Sequenz r defekt werden und danach die Gondelsequenz g beobachtet werden kann.

Überprüfen der Gondelsequenz

Bei den ersten drei Subtasks musst du überprüfen, ob eine Eingabesequenz einer möglichen Gondelsequenz entspricht. Die untenstehende Tabelle zeigt Beispiele von möglichen und von nicht möglichen Gondelsequenzen. Implementiere eine Funktion `valid`.

- `valid(n, inputSeq)`
  - `n`: die Länge der Eingabesequenz.
  - `inputSeq`: Array der Länge n; `inputSeq[i]` entspricht Element i der Eingabesequenz, für $0 \leq i \leq n-1$.
  - Entspricht die Eingabesequenz einer Gondelsequenz, gibt die Funktion 1 zurück, sonst 0.

Subtasks 1, 2, 3

Subtask	Punkte	n	`inputSeq`
1	5	$n \leq 100$	enthält jede Zahl von 1 bis n exakt einmal
2	5	$n \leq 100\,000$	$1 \leq$ `inputSeq[i]` $\leq n$
3	10	$n \leq 100\,000$	$1 \leq$ `inputSeq[i]` $\leq 250\,000$

Beispiele

Subtask	`inputSeq`	Rückgabewert	Bemerkung
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 kann nicht direkt vor 5 erscheinen
1	(4, 3, 2, 1)	0	4 kann nicht direkt vor 3 erscheinen
2	(1, 2, 3, 4, 5, 6, 5)	0	zwei Gondeln mit der Nummer 5
3	(2, 3, 4, 9, 6, 7, 1)	1	Ersatzsequenz (5, 8)
3	(10, 4, 3, 11, 12)	0	4 kann nicht direkt vor 3 erscheinen

"Replacement"-Sequenz

In den nächsten drei Subtasks konstruierst du jeweils eine mögliche "Replacement"-Sequenz, die eine gegebene Gondelsequenz erzeugt. Jede entsprechende "Replacement"-Sequenz wird akzeptiert werden. Implementiere eine Funktion `replacement`.

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n`: die Länge der Gondelsequenz.
  - `gondolaSeq`: Array der Länge n; `gondolaSeq` ist garantiert eine Gondelsequenz, und `gondolaSeq[i]` ist Element i der Sequenz, für $0 \leq i \leq n-1$.

- Die Funktion soll die Länge l der "Replacement"-Sequenz zurückgeben.
- `replacementSeq`: Array, gross genug um die "Replacement"-Sequenz zu speichern; die Rückgabe deiner Sequenz erfolgt durch Speichern von Element i deiner "Replacement"-Sequenz in `replacementSeq[i]`, für $0 \leq i \leq l - 1$.

Subtasks 4, 5, 6

Subtask	Punkte	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]` $\leq 250\,000$

Beispiele

Subtask	`gondolaSeq`	Rückgabewert	`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)

Anzahl an "Replacement"-Sequenzen

In den nächsten vier Subtasks ermittelst du jeweils die Anzahl an möglichen "Replacement"-Sequenzen für eine gegebene Sequenz (welche eine Gondelsequenz darstellen kann oder nicht), Modulo $1\,000\,000\,009$. Implementiere eine Funktion `countReplacement`.

- `countReplacement(n, inputSeq)`
  - `n`: die Länge der Eingabesequenz.
  - `inputSeq`: Array der Länge n; `inputSeq[i]` ist Element i der Eingabesequenz, für $0 \leq i \leq n - 1$.
  - Ist die Eingabesequenz eine Gondelsequenz, ermittle die - eventuell extrem grosse - Anzahl an "Replacement"-Sequenzen, welche diese Gondelsequenz erzeugt, *und gib diese Anzahl Modulo* $1\,000\,000\,009$ *zurück*. Ist die Eingabesequenz keine Gondelsequenz, gibt die Funktion 0 zurück. Ist die Eingabesequenz eine Gondelsequenz in der keine Gondeln defekt sind, gibt die Funktion 1 zurück.

Subtasks 7, 8, 9, 10

Subtask	Punkte	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, und mindestens $n - 3$ der ursprünglichen Gondeln $1, \ldots, n$ sind nicht defekt.

Subtask	Punkte	n	inputSeq
9	15	$n \leq 100\,000$	$1 \leq$ inputSeq[i] $\leq 250\,000$
10	10	$n \leq 100\,000$	$1 \leq$ inputSeq[i] $\leq 1\,000\,000\,000$

Beispiele

Subtask	inputSeq	Rückgabewert	"Replacement"-Sequenz
7	(1, 2, 7, 6)	2	(3, 4, 5) oder (4, 5, 3)
8	(2, 3, 4, 12, 6, 7, 1)	1	(5, 8, 9, 10, 11)
9	(4, 7, 4, 7)	0	inputSeq ist keine Gondelsequenz
10	(3, 4)	2	(1, 2) oder (2, 1)

Implementierungsdetails

Gib exakt eine Datei ab. Gib ihr einen der Namen gondola.c, gondola.cpp oder gondola.pas. Diese Datei muss alle drei oben beschriebenen Unterprogramme implementieren (sogar, wenn du planst, nur einige Subtasks zu implementieren). Gebrauche eine der folgenden Signaturen. Du musst auch die Headerdatei gondola.h in die C/C++ Implementation einfügen.

Programme in C/C++

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

Programme in 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

Der Sample-Grader liest die Eingabe im folgenden Format:

- Zeile 1: T, die Nummer des Subtasks, die dein Programm zu lösen versucht ($1 \leq T \leq 10$).
- Zeile 2: n, die Länge der Eingabesequenz.
- Zeile 3: Wenn T 4, 5, oder 6 ist, enthält diese Zeile gondolaSeq[0], ..., gondolaSeq[n-1]. Ansonsten enthält sie inputSeq[0], ..., inputSeq[n-1].