Gondola

Mao-Kong Gondola is a famous attraction in Taipei. The gondola system consists of a circular rail, a single station, and n gondolas numbered consecutively from 1 to n running around the rail in a fixed direction. After gondola i passes the station, the next gondola to pass the station will be gondola $i + 1$ if $i < n$, or gondola 1 if $i = n$.

Gondolas may break down. Luckily we have an infinite supply of spare gondolas, which are numbered $n + 1$, $n + 2$, and so on. When a gondola breaks down we replace it (in the same position on the track) with the first available spare gondola, that is, the one with the lowest number. For example, if there are five gondolas and gondola 1 breaks down, then we will replace it with gondola 6.

You like to stand at the station and watch the gondolas as they pass by. A *gondola sequence* is a sequence of n numbers of gondolas that pass the station. It is possible that one or more gondolas broke down (and were replaced) before you arrived, but none of the gondolas break down while you are watching.

Note that the same configuration of gondolas on the rail can give multiple gondola sequences, depending on which gondola passes first when you arrive at the station. For example, if none of the gondolas have broken down then both (2, 3, 4, 5, 1) and (4, 5, 1, 2, 3) are possible gondola sequences, but (4, 3, 2, 5, 1) is not (because the gondolas appear in the wrong order).

If gondola 1 breaks down, then we might now observe the gondola sequence (4, 5, 6, 2, 3). If gondola 4 breaks down next, we replace it with gondola 7 and we might observe the gondola sequence (6, 2, 3, 7, 5). If gondola 7 breaks down after this, we replace it with gondola 8 and we may now observe the gondola sequence (3, 8, 5, 6, 2).

broken gondola	new gondola	possible gondola sequence
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

A *replacement sequence* is a sequence consisting of the numbers of the gondolas that have broken down, in the order in which they break down. In the previous example the replacement sequence is (1, 4, 7). A replacement sequence r *produces* a gondola sequence g if, after gondolas break down according to the replacement sequence r, the gondola sequence g may be observed.

Gondola Sequence Checking

In the first three subtasks you must check whether an input sequence is a gondola sequence. See the table below for examples of sequences that are and are not gondola sequences. You need to implement a function `valid`.

- `valid(n, inputSeq)`
  - `n`: the length of the input sequence.
  - `inputSeq`: array of length n; `inputSeq[i]` is element i of the input sequence, for $0 \leq i \leq n-1$.
  - The function should return 1 if the input sequence is a gondola sequence, or 0 otherwise.

Subtasks 1, 2, 3

subtask	points	n	inputSeq
1	5	$n \leq 100$	has each number from 1 to n exactly once
2	5	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq n$
3	10	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 250,000$

Examples

subtask	inputSeq	return value	note
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 cannot appear just before 5
1	(4, 3, 2, 1)	0	4 cannot appear just before 3
2	(1, 2, 3, 4, 5, 6, 5)	0	two gondolas numbered 5
3	(2, 3, 4, 9, 6, 7, 1)	1	replacement sequence (5, 8)
3	(10, 4, 3, 11, 12)	0	4 cannot appear just before 3

Replacement Sequence

In the next three subtasks you must construct a possible replacement sequence that produces a given gondola sequence. Any such replacement sequence will be accepted. You need to implement a function `replacement`.

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n` is the length of the gondola sequence.
  - `gondolaSeq`: array of length n; `gondolaSeq` is guaranteed to be a gondola sequence, and `gondolaSeq[i]` is element i of the sequence, for $0 \leq i \leq n-1$.
  - The function should return l, the length of the replacement sequence.
  - `replacementSeq`: array that is sufficiently large to store the replacement sequence; you should return your sequence by placing element i of your replacement sequence into

`replacementSeq[i]`, for $0 \leq i \leq l-1$.

Subtasks 4, 5, 6

subtask	points	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]` $\leq 250,000$

Examples

subtask	`gondolaSeq`	return value	`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)

Count Replacement Sequences

In the next four subtasks you must count the number of possible replacement sequences that produce a given sequence (which may or may not be a gondola sequence), modulo **1,000,000,009**. You need to implement a function `countReplacement`.

- `countReplacement(n, inputSeq)`
  - `n`: the length of the input sequence.
  - `inputSeq`: array of length n; `inputSeq[i]` is element i of the input sequence, for $0 \leq i \leq n-1$.
  - If the input sequence is a gondola sequence, then count the number of replacement sequences that produce this gondola sequence (which could be extremely large), *and return this number modulo **1,000,000,009***. If the input sequence is not a gondola sequence, the function should return 0. If the input sequence is a gondola sequence but no gondolas broke down, the function should return 1.

Subtasks 7, 8, 9, 10

subtask	points	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, and at least $n-3$ of the initial gondolas $1, \ldots, n$ did not break down.
9	15	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 250,000$
10	10	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 1,000,000,000$

Examples

subtask	inputSeq	return value	replacement sequence
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` is not a gondola sequence
10	(3, 4)	2	(1, 2) or (2, 1)

Implementation details

You have to submit exactly one file, called `gondola.c`, `gondola.cpp` or `gondola.pas`. This file should implement all three subprograms described above (even if you only plan to solve some of the subtasks), using the following signatures. You also need to include a header file `gondola.h` for C/C++ implementation.

C/C++ programs

```
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

The sample grader reads the input in the following format:

- line 1: `T`, the subtask number your program intends to solve ($\mathbf{1 \leq T \leq 10}$).
- line 2: `n`, the length of the input sequence.
- line 3: If `T` is 4, 5, or 6, this line contains `gondolaSeq[0]`, ..., `gondolaSeq[n-1]`. Otherwise this line contains `inputSeq[0]`, ..., `inputSeq[n-1]`.