缆车（Gondola）

猫空缆车（Mao-Kong Gondola）是台北市的一个著名景点。这个缆车系统包括一个环形轨道、一个缆车站和n个编号为1到n的缆车。这些缆车以固定的方向在轨道上循环运行。在缆车i经过缆车站之后，下一个经过缆车站的缆车将会是$i+1$（$i<n$时），或者是缆车1（$i=n$时）。

缆车可能会发生故障。幸运的是，我们有无限多个后备的空闲缆车，其编号依次为$n+1$, $n+2$等等。当某个缆车发生故障时，我们会在轨道上的同一位置用最前一个空闲缆车替换它，也就是说，编号最小的空闲缆车。举个例子，如果当前有五辆缆车而缆车1发生了故障，那么我们将用缆车6来替换它。

你喜欢去缆车站上观察缆车过站。一个*缆车序列（gondola sequence）*是由缆车过站次序形成的n个缆车编号的序列。在你到达缆车站之前，有可能会有一到多个缆车发生故障（并且被替换掉），但是在你观察过程中是不会有缆车发生故障的。

注意，在轨道上的相同一组缆车，有可能给出多种缆车序列，这取决于当你到缆车站时哪辆缆车最先过站。举个例子，如果没有任何缆车发生故障，那么(2, 3, 4, 5, 1)和(4, 5, 1, 2, 3)都可能是缆车序列，但是(4, 3, 2, 5, 1)不可能是（因为缆车的次序有误）。

如果缆车1发生故障，那么我们可能会观察到缆车序列(4, 5, 6, 2, 3)。如果接着缆车4发生故障而我们用缆车7替换它，就有可能会观察到缆车序列(6, 2, 3, 7, 5)。如果缆车7在此后发生故障而我们用缆车8替换它，那么现在就有可能会观察到缆车序列(3, 8, 5, 6, 2)。

故障缆车	新缆车	可能的缆车序列
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

一个*替换序列（replacement sequence）*是一个由故障缆车编号组成的序列，其次序与故障发生次序相同。在前面的例子中，替换序列是(1, 4, 7)。如果一个替换序列r对应的故障发生后，我们由此有可能观察到缆车序列g，就称替换序列r*生成*缆车序列g。

缆车序列检查

在前三个子任务中，你必须检查某个输入序列是否是一个缆车序列。下表举例说明了哪些序列是缆车序列而哪些不是。你需要实现一个函数valid。

- `valid(n, inputSeq)`
  - n: 输入序列的长度。
  - `inputSeq`: 大小为n的数组；`inputSeq[i]`是输入序列的第i个元素，其中

$0 \le i \le n-1$。

- 当输入序列是一个缆车序列时，函数应返回1，否则返回0。

子任务1, 2, 3

子任务	分值	n	inputSeq
1	5	$n \le 100$	从1到n的数字恰好各出现一次
2	5	$n \le 100,000$	$1 \le$ inputSeq[i] $\le n$
3	10	$n \le 100,000$	$1 \le$ inputSeq[i] $\le 250,000$

例子

子任务	inputSeq	返回值	备注
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不能恰好出现在5之前
1	(4, 3, 2, 1)	0	4不能恰好出现在3之前
2	(1, 2, 3, 4, 5, 6, 5)	0	有两个缆车编号都是5
3	(2, 3, 4, 9, 6, 7, 1)	1	替换序列是(5, 8)
3	(10, 4, 3, 11, 12)	0	4不能恰好出现在3之前

替换序列

在接下来的三个子任务中，你必须构造一个能够生成给定缆车序列的可能的替换序列。满足条件的任意替换序列都可以。你需要实现一个函数replacement。

- replacement(n, gondolaSeq, replacementSeq)
  - n是缆车序列的长度。
  - gondolaSeq: 大小为n的数组；gondolaSeq保证是一个缆车序列，而gondolaSeq[i]是序列中的第i个元素，这里$0 \le i \le n-1$。
  - 函数应返回替换序列的长度l。
  - replacementSeq: 一个足够大的能存下替换序列的数组；你应当将替换序列中的第i个元素存放到replacementSeq[i]做为返回结果，这里$0 \le i \le l-1$。

子任务4、5、6

子任务	分值	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] $\leq 250,000$

例子

子任务	gondolaSeq	返回值	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)

替换序列计数

在接下来的四个子任务中，你必须计算所有能够生成给定序列（有可能是缆车序列，也有可能不是）的可能替换序列的数目，并将其对**1,000,000,009**取模。 你需要实现一个函数countReplacement。

- countReplacement(n, inputSeq)
  - n: 输入序列的长度。
  - inputSeq: 大小为n的数组；inputSeq[i]是输入序列的第i个元素，这里 $0 \leq i \leq n - 1$。
  - 如果输入序列是一个缆车序列，则计算能够生成该缆车序列的可能的替换序列的数目（有可能会非常大），*然后将该数值对**1,000,000,009**取模做为返回值*。如果输入序列不是一个缆车序列，函数应返回0。如果输入序列是一个缆车序列，但是没有缆车发生故障，函数应返回1。

子任务7、8、9、10

子任务	分值	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, 初始缆车$1, \ldots, n$中至少有$n - 3$个不会发生故障。
9	15	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 250,000$
10	10	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 1,000,000,000$

例子

子任务	inputSeq	返回值	替换序列
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不是一个缆车序列
10	(3, 4)	2	(1, 2) or (2, 1)

实现细节

你只能提交一个文件，名为gondola.c、gondola.cpp或者gondola.pas。在该文件中应实现前面所述的三个函数（即便你只想解决其中的部分子任务，也要给出全部函数），并遵循下述命名与接口。对于C/C++程序，你还需要包含头文件gondola.h。

C/C++程序

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

Pascal程序

```
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;
```

评测方式

评测系统将会读入如下格式的输入数据：

- 第1行: T，你的程序需要解决的子任务编号($1 \leq T \leq 10$)。
- 第2行: n，输入序列的长度。
- 第3行: 如果T是4、5或者6，此行包含gondolaSeq[0], ..., gondolaSeq[n-1]。否则此行包含inputSeq[0], ..., inputSeq[n-1]。