ゴンドラ (Gondola)

Mao-Kong Gondola は台北で有名なゴンドラのアトラクションである．Mao-Kong Gondola のゴンドラ設備は，円環状のレール 1 本と，駅 1 個と，1 から n までの番号がつけられた n 個のゴンドラで構成される．ゴンドラ i が駅を通過した次には，$i < n$ ならばゴンドラ $i+1$ が，$i = n$ ならばゴンドラ 1 が来ることになっている．

ゴンドラは壊れることがある．幸いにも，壊れたゴンドラの代わりとなるゴンドラがいくらでもある．これらのゴンドラには，$n+1, n+2, \ldots$ と番号がつけられている．ゴンドラが壊れた際には壊れたゴンドラがあった場所と同じ場所に，代わりとなる未使用のゴンドラのうち最小の番号がつけられたゴンドラが設置される．例えば，最初に 5 つのゴンドラがあってゴンドラ 1 が壊れた場合，ゴンドラ 1 はゴンドラ 6 に置き換えられる．

あなたは駅に立って通りすぎるゴンドラを観察するのが好きである．**ゴンドラ列** (gondola sequence) とは，駅を通過する n 個のゴンドラの番号からなる数列である．あなたが駅に到着するよりも前に 1 つ以上のゴンドラが壊れ，別のゴンドラに置き換えられていることがありうる．しかし，あなたがゴンドラを観察している間にはゴンドラは 1 つも壊れない．

あなたが最初に観察するゴンドラ次第では，レール上のゴンドラの配置が同じでも，ゴンドラ列が異なる可能性があることに注意せよ．例えば，どのゴンドラも壊れていない状態を考えると，この状態では $(2, 3, 4, 5, 1)$ も $(4, 5, 1, 2, 3)$ もゴンドラ列として考えられる．一方で，$(4, 3, 2, 5, 1)$ はゴンドラが正しくない順番で現れているのでゴンドラ列としては考えられない．

この状態からゴンドラ 1 が壊れた場合は，現在のゴンドラ列として，例えば $(4, 5, 6, 2, 3)$ が考えられるようになる．次にゴンドラ 4 が壊れた場合は，ゴンドラ 4 はゴンドラ 7 に置き換えられ，現在のゴンドラ列として，例えば $(6, 2, 3, 7, 5)$ が考えられるようになる．この後にゴンドラ 7 が壊れた場合は，ゴンドラ 8 に置き換えられ，現在のゴンドラ列として，例えば $(3, 8, 5, 6, 2)$ が考えられるようになる．

壊れたゴンドラ	新しいゴンドラ	置き換え後に考えられるゴンドラ列の 1 つ
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

置き換え列 (replacement sequence) とは，これまでに壊れたゴンドラの番号を，ゴンドラが壊れた順番に並べた数列であると定義する．上記の例において，置き換え列は $(1, 4, 7)$ となる．置き換え列 r がゴンドラ列 g を**生成する** (produce) とは，初期状態からゴンドラが置き換え列 r に従って壊れた後に観察されるゴンドラ列としてゴンドラ列 g が考えられることであると定義する．

ゴンドラ列の検証 (Gondola Sequence Checking)

最初の 3 つの小課題では，あなたは入力として与えられる数列がゴンドラ列であるかどうかを検証しなければならない．下の表はゴンドラ列の例とゴンドラ列でない数列の例をいくつか示す．関数 `valid` を実装せよ．

- `valid(n, inputSeq)`
  - `n`： 入力として与えられる数列の長さ．
  - `inputSeq`： サイズ n の配列であり，`inputSeq[i]` は入力として与えられる数列の要素 i である $(0 \leqq i \leqq n-1)$.
  - この関数は，入力として与えられる数列がゴンドラ列なら 1 を，そうでないなら 0 を返すこと．

小課題 1, 2, 3 (Subtasks 1, 2, 3)

小課題	得点	n	`inputSeq`
1	5	$n \leqq 100$	1 以上 n 以下の整数がちょうど 1 回ずつ現れる．
2	5	$n \leqq 100,000$	$1 \leqq$ `inputSeq[i]` $\leqq n$
3	10	$n \leqq 100,000$	$1 \leqq$ `inputSeq[i]` $\leqq 250,000$

例 (Examples)

小課題	`inputSeq`	戻り値	備考
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	ゴンドラ 5 の直前にゴンドラ 1 が来ることはない．
1	(4, 3, 2, 1)	0	ゴンドラ 3 の直前にゴンドラ 4 が来ることはない．
2	(1, 2, 3, 4, 5, 6, 5)	0	ゴンドラ 5 が 2 個現れている．
3	(2, 3, 4, 9, 6, 7, 1)	1	置き換え列が (5, 8) となる場合が考えられる．
3	(10, 4, 3, 11, 12)	0	ゴンドラ 3 の直前にゴンドラ 4 が来ることはない．

置き換え列 (Replacement Sequence)

続く 3 つの小課題では，与えられたゴンドラ列を生成する置き換え列を 1 つ構成しなければならない．条件を満たす置き換え列は，いずれも正解と判定される．関数 `replacement` を実装せよ．

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n`： ゴンドラ列の長さ．
  - `gondolaSeq`： サイズ n の配列であり，これはゴンドラ列であることが保証されている．`gondolaSeq[i]` は入力として与えられるゴンドラ列の要素 i である $(0 \leqq i \leqq n-1)$.
  - この関数は，置き換え列の長さ l を返すこと．
  - `replacementSeq`: 置き換え列を格納するのに充分なサイズを持った配列であり，あなたは，あなたが構成した置き換え列の要素 i を `replacementSeq[i]` に代入しなければならない $(0 \leqq i \leqq l-1)$.

小課題 4, 5, 6 (Subtasks 4, 5, 6)

小課題	得点	n	`gondolaSeq`
4	5	$n \leqq 100$	$1 \leqq$ `gondolaSeq[i]` $\leqq n+1$
5	10	$n \leqq 1,000$	$1 \leqq$ `gondolaSeq[i]` $\leqq 5,000$
6	20	$n \leqq 100,000$	$1 \leqq$ `gondolaSeq[i]` $\leqq 250,000$

例 (Examples)

小課題	`gondolaSeq`	戻り値	`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)

置き換え列の個数 (Count Replacement Sequences)

続く 4 つの小課題では，あなたは与えられた数列を生成する置き換え列の個数を数え，それを **1,000,000,009** で割った余りを計算しなければならない．これらの小課題では与えられる数列がゴンドラ列であるとは限らない．関数 `countReplacement` を実装せよ．

- `countReplacement(n, inputSeq)`
  - `n`： 入力として与えられる数列の長さ．
  - `inputSeq`： サイズ n の配列であり，`inputSeq[i]` は入力として与えられる数列の要素 i である $(0 \leqq i \leqq n-1)$.
  - 入力として与えられる数列がゴンドラ列である場合は，このゴンドラ列を生成する置き換え列の個数 (この値は非常に大きくなるかもしれない) を数え，**置き換え列の個数を 1,000,000,009 で割った余り**を返すこと．入力として与えられる数列がゴンドラ列でない場合は，0 を返さなければならない．入力として与えられる数列がゴンドラ列であるが，どのゴンドラも壊れていない場合は，1 を返さなければならない．

小課題 7, 8, 9, 10 (Subtasks 7, 8, 9, 10)

小課題	得点	n	`inputSeq`
7	5	$4 \leqq n \leqq 50$	$1 \leqq$ `inputSeq[i]` $\leqq n+3$
8	15	$4 \leqq n \leqq 50$	$1 \leqq$ `inputSeq[i]` $\leqq 100$ であり，最初からあるゴンドラ (ゴンドラ 1 からゴンドラ n まで) のうち，少なくとも $n-3$ 個のゴンドラは壊れていない．
9	15	$n \leqq 100,000$	$1 \leqq$ `inputSeq[i]` $\leqq 250,000$
10	10	$n \leqq 100,000$	$1 \leqq$ `inputSeq[i]` $\leqq 1,000,000,000$

例 (Examples)

小課題	`inputSeq`	戻り値	考えられる置き換え列
7	(1, 2, 7, 6)	2	(3, 4, 5) あるいは (4, 5, 3)
8	(2, 3, 4, 12, 6, 7, 1)	1	(5, 8, 9, 10, 11)
9	(4, 7, 4, 7)	0	`inputSeq` はゴンドラ列ではない．
10	(3, 4)	2	(1, 2) あるいは (2, 1)

実装の詳細 (Implementation details)

1 つのファイルを提出せよ．提出するファイルの名前は gondola.c, gondola.cpp, gondola.pas のいずれかである．このファイルは，(たとえあなたが一部の小課題のみを解く場合でも) 課題で指定された 3 つのサブプログラムすべてを以下のシグネチャを用いて実装しなければならない．C/C++ のプログラムにおいては，gondola.h をインクルード (include) する必要がある．

C/C++ プログラム (C/C++ programs)

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

Pascal プログラム (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;
```

採点プログラムのサンプル (Sample grader)

採点プログラムのサンプルは，以下のフォーマットで入力を読み込む：

- 1 行目： T．T ($1 \leqq T \leqq 10$) はあなたのプログラムが解答しようとする小課題の番号を表す．
- 2 行目： n．n は入力で与えられる数列の長さを表す．
- 3 行目： T が 4, 5, 6 のいずれかである場合，この行は gondolaSeq[0] , . . . , gondolaSeq[n-1] を含む．それ以外の場合，この行は inputSeq[0] , . . . , inputSeq[n-1] を含む．