貓空纜車

貓空纜車是台北著名的景點。二十二世紀的台北市政府，重建貓空觀光纜車系統。新的貓纜系統被設計爲一個圓形的單向軌道以及一個單一的車站，所有的纜車在單向軌道上沿著同一個方向進行。軌道上有n個纜車，編號爲 1 到 n。一開始的時候，在車站月台邊，看到纜車i通過月台，則下一個通過月台的將會是纜車$i+1$ $(i < n)$，但是纜車n例外，因爲纜車n後面的纜車是纜車1。

纜車運行難免會故障，不過幸運地是二十二世紀的台北市政府非常有錢，我們擁有無限量的備用纜車，編號爲 $n+1, n+2$,...,以此類推。當有纜車故障時，捷運局會立刻用最神奇的方式直接將故障的纜車卸下來，並在該位置換上備用纜車中號碼最小的那一輛纜車。舉例來說，假設有5輛纜車在軌道上運行，纜車1發生故障，捷運局將會使用纜車6來替換纜車1。

你站在車站月台邊，觀看纜車依序通過。所謂的"纜車順序"是指n輛纜車通過車站的順序。我們假設在你到達月台邊之前，一輛或多輛纜車可能發生故障並且已經被替換完成，但是假設當你在月台邊觀看"纜車順序"時，並沒有纜車會發生故障。

因爲新的貓空纜車系統是圓形單向軌道，所以在固定的纜車運行下，你可能看到不同的"纜車順序"，這個順序取決於你到達月台邊看到的第一個纜車號碼。舉例來說，如果一開始運行時，都沒有纜車故障，那你看到的"纜車順序"可能是(2,3,4,5,1)，也可能是(4,5,1,2,3)，但是你就不可能看到(4,3,2,5,1)，因爲纜車的循環順序不對。

如果纜車1故障，則你可能觀察到(4,5,6,2,3)；假設接下來纜車4發生故障，捷運局就會用纜車7來替代，那你可能會觀察到(6,2,3,7,5)；如果這時候，剛換上去的纜車7也發生故障，則捷運局會使用纜車8來替換，這時你可能觀察到(3,8,5,6,2)。

故障的纜車編號	拿來替換的備用纜車編號	可能觀察到的"纜車順序"
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

所謂"故障順序"是指發生故障的纜車順序(依照故障時間先後排序)。在剛剛的例子中，"故障順序"是(1,4,7)。纜車開始運行後，對於故障的發生，捷運局依照 "故障順序r" 進行一一排解，在排解之後，你可能觀察到"纜車順序g"；對於這樣的狀況，我們稱 "故障順序r" 產生了"纜車順序g"。

纜車順序檢查

在前三個子任務中，你必須檢查輸入的數列是否爲一個"纜車順序"。下表列出了是否爲"纜車順序"的一些例子。你必須實作`valid`函式來解決這個問題。

- valid(n, inputSeq)
  - n: 輸入數列的長度。
  - inputSeq: 長度爲n的數列； inputSeq[i] 是輸入的第 i 個數字，$0 \le i \le n-1$。
  - 如果輸入的數列是一個"纜車順序"，請回傳**1**，如果不是，請回傳0。

子任務 1, 2, 3

子任務	分數	n	inputSeq
1	5	$n \le 100$	1 到 n 恰各出現一次
2	5	$n \le 100,000$	$1 \le$ inputSeq[i] $\le n$
3	10	$n \le 100,000$	$1 \le$ inputSeq[i] $\le 250,000$

範例

子任務	inputSeq	回傳值	說明
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	纜車**1**不可能恰好是纜車**5**的前一輛
1	(4, 3, 2, 1)	0	纜車**4**不可能恰好是纜車**3**的前一輛
2	(1, 2, 3, 4, 5, 6, 5)	0	數列中有兩個纜車**5**
3	(2, 3, 4, 9, 6, 7, 1)	1	"故障順序"爲(5, 8)
3	(10, 4, 3, 11, 12)	0	纜車**4**不可能恰好是纜車**3**的前一輛

故障順序

接下來的3個子任務，在給定的"纜車順序"中，你必須建構可能的"故障順序"；任何可以產生給定的"纜車順序"的"故障順序"都可以被接受。你必須實作函式replacement。

- replacement(n, gondolaSeq, replacementSeq)
  - n 是gondolaSeq的長度，也就是"纜車順序"的長度。
  - gondolaSeq: 長度爲n的數列；gondolaSeq 保證是一個可以發生的"纜車順序"，gondolaSeq[i] 是該數列中第i個纜車編號，$0 \le i \le n-1$。
  - 本函式應該回傳"故障順序"的長度 l。
  - replacementSeq: 用來儲存"故障順序"的陣列，其長度很長，足夠儲存"故障順序"，你必須將你計算的"故障順序"放在這個陣列中，replacementSeq[i]是第i

個故障的纜車編號，$0 \leq i \leq l-1$。

子任務 4, 5, 6

子任務	分數	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] $\leq 250,000$

範例

子任務	gondolaSeq	回傳值	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)

計算可能的"故障順序"個數

在接下來的四個子任務中，給定一個數列，該數列可能是有效的"纜車順序"，也可能不是"纜車順序"，你必須實作函式countReplacement來計算產生這樣數列的"故障順序"有幾個。將該個數對**1,000,000,009**取餘數然後回傳。

- countReplacement(n, inputSeq)
  - n: 該輸入數列inputSeq的長度。
  - inputSeq: 長度爲n的數列; inputSeq[i] 是輸入數列的第 i個元素，$0 \leq i \leq n-1$。
  - 如果該輸入數列是一個"纜車順序"，則計算可以產生此"纜車順序"的"故障順序"的個數，這個數字可能很大，所以你必須回傳*這個數字對**1,000,000,009**的餘數*。如果輸入數列不是一個"纜車順序"，則回傳**0**，如果輸入數列是一個"纜車順序"，但是沒有任何纜車故障，則請回傳**1**。

子任務 7, 8, 9, 10

子任務	分數	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, 且一開始運作的纜車中$1,\ldots,n$，至少有$n-3$輛不曾發生故障。
9	15	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 250,000$
10	10	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 1,000,000,000$

範例

子任務	inputSeq	回傳值	故障順序
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 不是一個"纜車順序"
10	(3, 4)	2	(1, 2) or (2, 1)

實作細節

你必須依照你使用的語言繳交一個檔案，取名 gondola.c, gondola.cpp 或 gondola.pas。在這個檔案中，你必須實作上面提到的所有三個函式，就算你只是想要解決某些子任務，你也必須實作所有函式。這些函式必須使用下列的函式原型。如果你使用C/C++，請你引用標頭檔gondola.h。

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;
```

範例評分程式

範例評分程式將會讀取輸入資料如下。

- 第1行: T, 子任務編號 ($1 \leq T \leq 10$).
- 第2行: n, 輸入數列的長度。
- 第3行: 如果 T 是 4, 5, 或 6, 則本行將是 gondolaSeq[0], ..., gondolaSeq[n-1]。否則本行將是inputSeq[0], ..., inputSeq[n-1]。