Kabínková lanovka (Gondola)

Kabínková lanovka Mao-Kong je známou atrakciou v meste Taipei. Lanovku tvorí jedno do kruhu natiahnuté oceľové lano. Na lane je n kabínok. (Po anglicky sa kabínke lanovky povie "gondola", odtiaľ pochádza anglický názov úlohy.) Keď bola lanovka nová, kabínky mali za radom čísla od 1 po n. Kabínky na lane sa stále hýbu tým istým smerom dookola - a to tak, že bezprostredne po kabínke i prejde stanicou kabínka $i + 1$ (resp. kabínka 1 pre $i = n$).

Kabínky sa občas pokazia. Pokazenú kabínku treba vymeniť za novú. Našťastie máme neobmedzenú zásobu nových kabínok. Tie majú čísla $n + 1$, $n + 2$, atď. Vždy, keď treba vymeniť pokazenú kabínku, použijeme náhradnú s najmenším dostupným číslom. Napríklad ak bolo 5 kabínok a pokazila sa ľubovoľná z nich, nahradíme ju kabínkou 6.

Rád stávaš na stanici a čumíš na kabínky chodiace okolo teba. *Kabínková postupnosť* je postupnosť práve n čísel kabínok, a to v poradí, v ktorom kabínky jedna za druhou prešli stanicou. Je možné, že pred tým ako si sa začal pozerať boli niektoré kabínky vyššie popísaným spôsobom vymenené. Žiadne kabínky sa ale nepokazili počas toho, ako si sa na ne pozeral.

Všimni si, že konkrétnemu poradiu kabínok na lane môže zodpovedať viacero kabínkových postupností. Konkrétny vzhľad postupnosti závisí od toho, ktorú kabínku uvidíš prechádzať stanicou ako prvú. Ak napríklad máme lanovku s 5 kabínkami a ešte sa žiadna z nich nepokazila, tak:

- (2,3,4,5,1) je kabínková postupnosť
- (4,5,1,2,3) je tiež kabínková postupnosť
- (4,3,2,5,1) nie je kabínková postupnosť - v takomto poradí stanicou prechádzať nemohli

Nasleduje ďalší príklad a jeho zhrnutie v prehľadnej tabuľke.

Keby sa kabínka 1 pokazila, mohli by sme pozorovať napríklad kabínkovú postupnosť (4,5,6,2,3). Keby sa následne pokazila aj kabínka 4, nahradili by sme ju kabínkou 7. Po tejto oprave by bola jednou z postupností, ktoré môžeme pozorovať, kabínková postupnosť (6,2,3,7,5). A keby sa následne pokazila ešte aj kabínka 7, nahradili by sme ju kabínkou 8. Po tejto (v poradí tretej) oprave by sme mohli pozorovať postupnosť (3,8,5,6,2).

pokazená kabínka	nová kabínka	jedna z kabínkových postupností
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

Keď lanovku postavili, zobrali si údržbári čistý list papiera. Vždy, keď sa nejaká kabínka pokazila, zapísali si na papier (na koniec zoznamu čísel, ktoré tam už boli uvedené) číslo *pokazenej* kabínky. Zoznam čísel, ktorý takto vznikol, voláme *postupnosť opráv*. Vyššie uvedenému príkladu zodpovedá postupnosť opráv (1,4,7).

Hovoríme, že postupnosť opráv r *vedie ku kabínkovej postupnosti* p, ak je možné, že sa pokazili kabínky v poradí udávanom postupnosťou r a následne sme pozorovali postupnosť kabínok p.

Kontrola kabínkovej postupnosti

V prvých troch podúlohách máš napísať funkciu `valid`, ktorá dostane na vstupe postupnosť čísel a skontroluje, či je táto postupnosť kabínková.

- `valid(n, inputSeq)`
  - `n`: dĺžka zadanej postupnosti
  - `inputSeq`: pole s n prvkami; `inputSeq[i]` je i-ty prvok zadanej postupnosti (pričom indexujeme od 0).
  - Funkcia má vrátiť 1 ak je vstupná postupnosť kabínková a 0 ak nie je.

Podúlohy 1, 2, 3

podúloha	body	n	`inputSeq`
1	5	$n \leq 100$	obsahuje každé z čísel 1 až n práve raz
2	5	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq n$
3	10	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 250,000$

Príklad pre podúlohy 1, 2, 3

podúloha	`inputSeq`	výstup	poznámka
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 nemôže prejsť stanicou bezprostredne pred 5
1	(4, 3, 2, 1)	0	4 nemôže byť hneď pred 3
2	(1, 2, 3, 4, 5, 6, 5)	0	dve kabínky číslo 5
3	(2, 3, 4, 9, 6, 7, 1)	1	vedie k nej postupnosť opráv (5, 8)
3	(10, 4, 3, 11, 12)	0	4 nemôže byť hneď pred 3

Postupnosti opráv

V ďalších troch podúlohách máš napísať funkciu `replacement`, ktorá dostane na vstupe *platnú* kabínkovú postupnosť a na výstupe vráti *jednu ľubovoľnú* postupnosť opráv, ktorá ku nej vedie.

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n`: dĺžka zadanej postupnosti
  - `gondolaSeq`: pole s n prvkami; `gondolaSeq[i]` je i-ty prvok zadanej postupnosti (pričom indexujeme od 0). Je zaručené, že `gondolaSeq` popisuje kabínkovú postupnosť.

- Funkcia má ako návratovú hodnotu dať číslo ℓ: dĺžku postupnosti opráv, ktorú chcete vrátiť.
- Funkcia má tiež v premennej `replacementSeq` vrátiť pole dostatočnej dĺžky (teda dĺžky aspoň ℓ). Pre $0 \leq i \leq \ell - 1$ by `replacementSeq[i]` mal byť i-ty (indexujúc od 0) prvok postupnosti opráv, ktorú si našiel.

Podúlohy 4, 5, 6

podúloha	body	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]` $\leq 250,000$

Príklady pre podúlohy 4, 5, 6

podúloha	`gondolaSeq`	návratová hodnota	`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)

Počet postupností opráv

V posledných štyroch podúlohách musíš napísať funkciu `countReplacement`, ktorá pre danú *ľubovoľnú* (nie nutne kabínkovú) postupnosť nájde *počet všetkých možných* postupností opráv, ktoré k vstupnej postupnosti vedú. Keďže môže ísť o veľmi veľké číslo, vypočítať a vrátiť na výstupe treba jeho hodnotu **modulo 1,000,000,009** (teda $10^9 + 9$).

- `countReplacement(n, inputSeq)`
  - `n`: dĺžka zadanej postupnosti
  - `inputSeq`: pole s n prvkami; `inputSeq[i]` je i-ty prvok zadanej postupnosti (pričom indexujeme od 0).
  - Nech x je počet postupností opráv, ktoré vedú k postupnosti zo vstupu. (Hodnota x môže byť veľmi veľká.) Ako návratovú hodnotu vráťte hodnotu $x \bmod 1,000,000,009$. (Ak vstupná postupnosť nie je kabínkovou postupnosťou, správna návratová hodnota je 0. Ak je vstupná postupnosť kabínková a nič sa ešte nepokazilo, správna odpoveď je 1 lebo existuje jedna vyhovujúca postupnosť opráv: prázdna.)

Podúlohy 7, 8, 9, 10

podúloha	body	n	`inputSeq`
7	5	$4 \leq n \leq 50$	$1 \leq$ `inputSeq[i]` $\leq n + 3$

podúloha	body	n	inputSeq
8	15	$4 \leq n \leq 50$	$1 \leq$ inputSeq[i] ≤ 100 a platí, že aspoň $n - 3$ z pôvodných kabínok $1, \ldots, n$ sa nikdy nepokazilo.
9	15	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 250,000$
10	10	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 1,000,000,000$

Príklady pre podúlohy 7, 8, 9, 10

podúloha	inputSeq	výstup	postupnosti opráv
7	(1, 2, 7, 6)	2	(3, 4, 5) a (4, 5, 3)
8	(2, 3, 4, 12, 6, 7, 1)	1	(5, 8, 9, 10, 11)
9	(4, 7, 4, 7)	0	inputSeq nie je kabínková postupnosť
10	(3, 4)	2	(1, 2) a (2, 1)

Detaily implementácie

Odovzdávaš presne jeden súbor, nazvaný `gondola.c`, `gondola.cpp` alebo `gondola.pas`. V tomto súbore by mala byť implementovaná vyššie popísaná funkcia `findSample`. Musí mať hlavičku uvedenú nižšie. Ak programuješ v C/C++, tvoj súbor musí vložiť (`include`) súbor `gondola.h`.

C/C++

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

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;
```

Ukážkový grader

Ukážkový grader, ktorý máte k dispozícii, očakáva vstup v nasledovnom formáte:

- riadok 1: `T`, číslo podúlohy ktorú chceš riešiť ($1 \leq T \leq 10$).
- riadok 2: `n`, dĺžka vstupnej postupnosti
- riadok 3: vstupná postupnosť

(Ak `T` je 4, 5, alebo 6, vstupná postupnosť bude postupnosťou `gondolaSeq[0]`, ..., `gondolaSeq[n-1]`. Inak pôjde o postupnosť `inputSeq[0]`, ..., `inputSeq[n-1]`.)