# Gondola

Ivan-Kong Gondola je poznata turistička atrakcija u Taipei-u. Sistem gondola se sastoji od jedne kružne pruge, jedne stanice i $n$ gondola uzastopno numerisanih brojevima od 1 do $n$ koje se kreću u krug prugom u fiksnom smeru. Kada gondola $i$ prođe kroz stanicu, sledeća gondola koja će proći kroz stanicu će biti gondola $i+1$ ako je $i < n$, ili gondola 1 ako je $i = n$.

Nažalost, Ivana, gondol-menadžera, briga za gondole i zato se gondole mogu pokvariti. Srećom on na raspolaganju ima neograničen broj rezervnih gondola, numerisanih brojevima $n+1$, $n+2$, itd. Kada se neka gondola pokvari on je zameni (na istom mestu na pruzi) prvom dostupnom rezervnom gondolom, tj. onom neiskorišćenom rezervnom gondolom sa najmanjim brojem. Na primer, ako imamo 5 gondola i gondola 1 se pokvari, onda će ona biti zamenjena gondolom 6.

Vi uživate da stojite kraj stanice i posmatrate kako prolaze gondole. *Gondola-niz* je niz od $n$ brojeva gondola koje (redom, dok vi posmatrate) prolaze kroz stanicu. Moguće je da se jedna ili više gondola pokvarilo (i da su bile zamenjene) pre nego što ste stigli do stanice, ali je poznato da se nijedna gondola ne može pokvariti dok posmatrate.

Primetimo da ista konfiguracija gondola na pruzi može proizvesti više različitih gondola-nizova, u zavisnosti od toga koja gondola prođe prva kroz stanicu kada vi dodjete na stanicu. Na primer, ako se nije pokvarila nijedna gondola onda su i (2, 3, 4, 5, 1) i (4, 5, 1, 2, 3) mogući gondola-nizovi koje možete uočiti, ali (4, 3, 2, 5, 1) nije (jer se gondole pojavljuju u pogrešnom redosledu).

Ukoliko se gondola 1 pokvari, jedan od mogućih gondola-nizova je (4, 5, 6, 2, 3). Ukoliko je gondola 4 sledeća koja se pokvari, Ivan je menja gondolom 7 i onda je (6, 2, 3, 7, 5) jedan od mogućih gondola-nizova. Ukoliko se sada pokvari gondola 7, Ivan je menja gondolom 8 i sada je moguće da primetite gondola-niz (3, 8, 5, 6, 2).

| pokvarena gondola | nova gondola | mogući gondola-niz |
|---|---|---|
| 1 | 6 | (4, 5, 6, 2, 3) |
| 4 | 7 | (6, 2, 3, 7, 5) |
| 7 | 8 | (3, 8, 5, 6, 2) |

*Zamena-niz* je niz koji se sastoji od brojeva gondola koje su se pokvarile, u redosledu u kome su se pokvarile. U prethodnom primeru zamena-niz je (1, 4, 7). Zamena-niz $r$ *daje* gondola-niz $g$ ako, kada se gondole pokvare na način opisan u zamena-nizu $r$, možete uočiti gondola-niz $g$.

## Provera gondola-nizova

U prva tri podzadataka je potrebno proveriti da li je dati ulazni niz gondola-niz. Pogledajte tabelu ispod za primere nizova koji jesu i koji nisu gondola-nizovi. Morate implementirati funkciju `valid`.

- `valid(n, inputSeq)`
  - `n`: dužina ulaznog niza.
  - `inputSeq`: niz dužine $n$; `inputSeq[i]` je $i$-ti element ulaznog niza, za $0 \leq i \leq n-1$.
  - Funkcija treba da vrati 1 ako je ulazni niz gondola-niz, a 0 u suprotnom.

### Podzadaci 1, 2, 3

| podzadatak | poeni | $n$ | `inputSeq` |
|---|---|---|---|
| 1 | 5 | $n \leq 100$ | svaki broj od 1 do $n$ se pojavljuje tačno jednom |
| 2 | 5 | $n \leq 100,000$ | $1 \leq$ `inputSeq[i]` $\leq n$ |
| 3 | 10 | $n \leq 100,000$ | $1 \leq$ `inputSeq[i]` $\leq 250,000$ |

### Primeri

| podzadatak | `inputSeq` | izlaz f-je `valid` | komentar |
|---|---|---|---|
| 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 se ne može pojaviti neposredno pre 5 |
| 1 | (4, 3, 2, 1) | 0 | 4 se ne može pojaviti neposredno pre 3 |
| 2 | (1, 2, 3, 4, 5, 6, 5) | 0 | dve gondole sa brojem 5 |
| 3 | (2, 3, 4, 9, 6, 7, 1) | 1 | zamena-niz (5, 8) |
| 3 | (10, 4, 3, 11, 12) | 0 | 4 se ne može pojaviti neposredno pre 3 |

# Zamena-niz

U sledeća tri podzadataka je potrebno odrediti neki zamena-niz koji daje dati gondola-niz. Bilo koji validni zamena-niz će biti prihvaćen. Morate implementirati funkciju `replacement`.

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n` je dužina gondola-niza.
  - `gondolaSeq`: niz dužine $n$; garantuje se da je `gondolaSeq` gondola-niz, i `gondolaSeq[i]` je $i$-ti element niza, za $0 \leq i \leq n-1$.
  - Funkcija treba da vrati $l$, dužinu zamena-niza.
  - `replacementSeq`: niz koji je dovoljne veličine da se u njemu smesti zamena-niz; vi treba da vratite traženi zamena-niz tako što ćete staviti $i$-ti element vašeg zamena-niza u `replacementSeq[i]`, za $0 \leq i \leq l-1$.

### Podzadaci 4, 5, 6

| podzadatak | poeni | $n$ | 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]` $\leq 250,000$ |

### Primeri

| podzadatak | gondolaSeq | izlaz f-je replacement | 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) |

# Prebrojavanje zamena-nizova

U sledeća četiri podzadataka potrebno je izračunati broj mogućih zamena-nizova koji daju dati niz (koji može ali ne mora biti gondola-niz) po modulu **1,000,000,009**. Morate implementirati funkciju `countReplacement`.

- `countReplacement(n, inputSeq)`
  - `n`: dužina ulaznog niza.
  - `inputSeq`: niz dužine $n$; `inputSeq[i]` je $i$-ti element ulaznog niza, za $0 \leq i \leq n-1$.
  - Ako je ulazni niz gondola-niz, izračunajte broj mogućih zamena-nizova koji daju dati gondola-niz (ovaj broj može biti jako veliki) *i vratite ovaj broj po modulu **1,000,000,009***. Ako ulazni niz nije gondola-niz, funkcija treba da vrati 0. Ako je ulazni niz gondola-niz ali se nijedna gondola nije pokvarila, funkcija treba da vrati 1.

### Podzadaci 7, 8, 9, 10

| podzadatak | poeni | $n$ | 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]` $\leq 100$, i bar $n-3$ od početnih gondola $1, \ldots, n$ se nije pokvarilo. |
| 9 | 15 | $n \leq 100,000$ | $1 \leq$ `inputSeq[i]` $\leq 250,000$ |
| 10 | 10 | $n \leq 100,000$ | $1 \leq$ `inputSeq[i]` $\leq 1,000,000,000$ |

### Primeri

| podzadatak | `inputSeq` | izlaz f-je `countReplacement` | zamena-nizovi |
|---|---|---|---|
| 7 | (1, 2, 7, 6) | 2 | (3, 4, 5) ili (4, 5, 3) |
| 8 | (2, 3, 4, 12, 6, 7, 1) | 1 | (5, 8, 9, 10, 11) |
| 9 | (4, 7, 4, 7) | 0 | `inputSeq` nije gondola-niz |
| 10 | (3, 4) | 2 | (1, 2) ili (2, 1) |

# Detalji implementacije

Potrebno je da pošaljete tačno jedan fajl koji je potrebno nazvati `gondola.c`, `gondola.cpp` ili `gondola.pas`. Ovaj fajl treba da implementira sve tri funkcije opisane gore (čak i ako ne planirate da radite sve podzadatke), koristeći sledeće potpise. Takođe je potrebno include-ovati i header fajl `gondola.h` za C/C++ implementaciju.

## C/C++ programi

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

## Pascal programi

```
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;
```

## Opis grejdera

Grejder koji se nalazi na vašem računaru čita ulazne podatke u sledećem obliku:

- linija 1: `T`, redni broj podzadatka ($\mathbf{1 \leq T \leq 10}$).
- linija 2: `n`, dužina ulaznog niza.
- linija 3: Ako je `T` 4, 5, ili 6, ova linija sadži `gondolaSeq[0]`, ..., `gondolaSeq[n-1]`. U suprotnom, ova linija sadži `inputSeq[0]`, ..., `inputSeq[n-1]`.