Gondole (Gondola)

Gondole Mao-Kong należą do najpopularniejszych atrakcji Taipei. Ten system gondoli składa się z zamkniętej w pierścień szyny, pojedynczej stacji oraz n gondoli, ponumerowanych kolejno od 1 do n, które poruszają się w kółko po szynie w ustalonym kierunku. Po tym, jak gondola i przejedzie przez stację, kolejną gondolą na stacji jest gondola $i+1$, jeśli $i < n$, oraz gondola 1, jeśli $i = n$.

Gondole mogą się jednak psuć. Szczęśliwie dostępna jest nieograniczona liczba zapasowych gondoli, ponumerowanych kolejno $n+1$, $n+2$, itd. Gdy gondola zepsuje się, zostaje ona zastąpiona (dokładnie w tym samym miejscu szyny) przez pierwszą dostępną gondolę zapasową, tj. gondolę zapasową o najmniejszym dostępnym numerze. Przykładowo, jeśli w systemie jest pięć gondoli i gondola 1 popsuje się, zostanie ona zastąpiona gondolą 6.

Uwielbiasz spędzać czas, stojąc na stacji i oglądając kolejno przejeżdżające przez nią gondole. *Ciągiem gondolowym* nazywamy sekwencję n numerów gondoli, które przejeżdżają kolejno przez stację. Mogło się zdarzyć, że przed Twoim przyjazdem niektóre gondole zepsuły się (i zostały zastąpione przez gondole zapasowe), jednak żadna gondola nie psuje się, podczas gdy przebywasz i obserwujesz je na stacji.

Zauważ, że w zależności od tego, która gondola jako pierwsza wjeżdża na stację, ta sama kolejność gondoli na szynie może dać różne ciągi gondolowe. Dla przykładu, jeśli żadna gondola jeszcze się nie popsuła, zarówno (2, 3, 4, 5, 1), jak i (4, 5, 1, 2, 3) są możliwymi ciągami gondolowymi, natomiast (4, 3, 2, 5, 1) nie może być takim ciągiem (ponieważ gondole występują w złej kolejności).

Jeśli gondola 1 popsuje się, jako ciąg gondolowy możemy zaobserwować m.in. ciąg (4, 5, 6, 2, 3). Jeśli jako kolejna popsuje się gondola 4, zostanie ona zastąpiona przez gondolę 7, co może prowadzić do ciągu gondolowego (6, 2, 3, 7, 5). Jeśli dalej popsuje się gondola 7, zastąpi ją gondola 8, a końcowym ciągiem gondolowym może być np. (3, 8, 5, 6, 2).

popsuta gondola	nowa gondola	możliwy ciąg gondolowy
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

Ciągiem zastąpień nazywamy ciąg numerów kolejnych gondol, które uległy popsuciu. Ciągiem zastąpień dla powyższego przykładu będzie więc (1, 4, 7). Powiemy, że ciąg zastąpień r *prowadzi do* ciągu gondolowego g, jeśli po tym, jak popsują się kolejno gondole wymienione w ciągu r, możemy zaobserwować ciąg gondolowy g.

Sprawdzanie gondolowości ciągu

W trzech pierwszych podzadaniach Twój program powinien sprawdzić, czy dany ciąg może być ciągiem gondolowym. W poniższej tabeli znajdują się przykłady ciągów gondolowych oraz takich, które

nie mogą być ciągami gondolowymi. Napisz funkcję `valid`.

- `valid(n, inputSeq)`
  - `n`: długość ciągu.
  - `inputSeq`: tablica rozmiaru n; `inputSeq[i]` to i-ty element ciągu, dla $0 \leq i \leq n-1$.
  - Wynikiem funkcji powinno być 1, jeśli dany ciąg jest ciągiem gondolowym, a 0 w przeciwnym przypadku.

Podzadania 1, 2, 3

podzadanie	liczba punktów	n	`inputSeq`
1	5	$n \leq 100$	każda liczba od 1 do n występuje dokładnie raz
2	5	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq n$
3	10	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 250,000$

Przykłady

podzadanie	`inputSeq`	wynik funkcji	uwagi
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 nie może wystąpić tuż przed 5
1	(4, 3, 2, 1)	0	4 nie może wystąpić tuż przed 3
2	(1, 2, 3, 4, 5, 6, 5)	0	dwie gondole o numerze 5
3	(2, 3, 4, 9, 6, 7, 1)	1	ciąg zastąpień to (5, 8)
3	(10, 4, 3, 11, 12)	0	4 nie może wystąpić tuż przed 3

Wyznaczanie ciągu zastąpień

W trzech kolejnych podzadaniach Twój program powinien odtworzyć przykładowy ciąg zastąpień, który prowadzi do zadanego ciągu gondolowego. Jeśli istnieje wiele możliwych ciągów zastąpień, możesz wybrać dowolny z nich. Napisz funkcję `replacement`.

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n`: długość ciągu gondolowego.
  - `gondolaSeq`: tablica rozmiaru n; jest zagwarantowane, że `gondolaSeq` to ciąg gondolowy, a `gondolaSeq[i]` to i-ty element tego ciągu, dla $0 \leq i \leq n-1$.
  - Wynikiem funkcji powinno być l, to jest długość ciągu zastąpień.

- `replacementSeq`: tablica wystarczająco duża na to, by przechować cały ciąg zastąpień; funkcja powinna umieścić i-ty element ciągu zastąpień w polu `replacementSeq[i]`, dla $0 \leq i \leq l-1$.

Podzadania 4, 5, 6

podzadanie	liczba punktów	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]` $\leq 250,000$

Przykłady

podzadanie	`gondolaSeq`	wynik funkcji	`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)

Zliczanie ciągów zastąpień

W czterech ostatnich podzadaniach Twój program powinien wyznaczyć liczbę różnych ciągów zastąpień, które prowadzą do podanego ciągu (który może być lub nie być ciągiem gondolowym), modulo **1,000,000,009**. Napisz funkcję `countReplacement`.

- `countReplacement(n, inputSeq)`
  - `n`: długość ciągu.
  - `inputSeq`: tablica rozmiaru n; `inputSeq[i]` to i-ty element ciągu, dla $0 \leq i \leq n-1$.
  - Jeśli dany ciąg jest ciągiem gondolowym, funkcja powinna wyznaczyć liczbę ciągów zastąpień, które prowadzą do tego ciągu gondolowego (która może być ogromniasta), i *podać w wyniku tę liczbę modulo **1,000,000,009***. Jeśli dany ciąg nie jest ciągiem gondolowym, wynikiem funkcji powinno być 0. Jeśli dany ciąg jest ciągiem gondolowym odpowiadającym sytuacji, że żadna gondola nie popsuła się, wynikiem funkcji powinno być 1.

Podzadania 7, 8, 9, 10

podzadanie	liczba punktów	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 i co najmniej $n-3$ spośród gondoli $1, \ldots, n$ nie popsuło się.

podzadanie	liczba punktów	n	inputSeq
9	15	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 250,000$
10	10	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 1,000,000,000$

Przykłady

podzadanie	inputSeq	wynik funkcji	ciąg zastąpień
7	(1, 2, 7, 6)	2	(3, 4, 5) lub (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 jest ciągiem gondolowym
10	(3, 4)	2	(1, 2) lub (2, 1)

Implementacja

Powinieneś zgłosić dokładnie jeden plik o nazwie gondola.c, gondola.cpp lub gondola.pas. W tym pliku powinna znaleźć się implementacja wszystkich trzech funkcji opisanych powyżej (nawet jeśli planujesz rozwiązać tylko niektóre podzadania), o następujących sygnaturach. W przypadku programu w C/C++ powinieneś także załączyć (*include*) plik nagłówkowy gondola.h.

Programy w C/C++

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

Programy w Pascalu

```
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;
```

Przykładowy program sprawdzający

Przykładowy program sprawdzający wczytuje dane w następującym formacie:

- wiersz 1: T, numer podzadania, które Twój program ma rozwiązać ($1 \leq T \leq 10$).
- wiersz 2: n, długość ciągu wejściowego.
- wiersz 3: Jeśli T jest równe 4, 5 lub 6, wiersz ten zawiera gondolaSeq[0], ..., gondolaSeq[n-1]. W przeciwnym razie wiersz ten zawiera inputSeq[0], ..., inputSeq[n-1].