Gondola

Mao-Kong Gondola är en känd turistattraktion i Taipei. Gondolsystemet består av en cirkulär räls, en enda station, och n gondolers numrerade från 1 till n som åker runt rälsen i en viss fixerad riktning. Gondolerna åker i följande ordning: efter att gondol i passerar stationen så passerar gondol $i + 1$ (om $i < n$), annars (om $i = n$) så passerar gondol 1.

Gondoler kan gå sönder. Lyckligtvis så har vi ett oändligt antal reservgondoler, som är numrerade $n + 1$, $n + 2$, och så vidare. När en gondol går sönder så ersätter vi den (vid samma position på rälsen som den gick sönder vid) med första tillgängliga reservgondol, d.v.s. den med minsta möjliga nummer. Till exempel, om det finns fem gondoler och gondol nummer 1 går sönder så kommer vi att ersätta den med gondol nummer 6.

Du gillar att stå och drägga vid stationen och titta på gondolerna när de åker förbi. En *gondolsekvens* är en serie n heltal som beskriver en sekvens av gondoler som passerar stationen. Det är möjligt att en eller flera gondoler går sönder (och blir ersatta) innan du anländer till stationen, men ingen av gondolerna går sönder medan du tittar.

Notera att konfigurationen av gondoler på rälsen kan ge ett flertal gondolsekvenser, beroende på vilken gondol som passerar först efter att du anlänt till stationen och börjat tittandet. Till exempel, om ingen av gondolerna har gått sönder så är (2, 3, 4, 5, 1) och (4, 5, 1, 2, 3) båda möjliga gondolsekvenser, men (4, 3, 2, 5, 1) är det inte (eftersom gondolerna förekommer i en ogiltig ordning).

Om gondol nummer 1 går sönder så kan vi observera gondolsekvensen (4, 5, 6, 2, 3). Om gondol nummer 4 sedan går sönder så ersätter vi den med gondol nummer 7 och vi observerar sekvensen (6, 2, 3, 7, 5). Om gondol nummer 7 sedan går sönder efter detta så ersätter vi den med gondol nummer 8 och vi ser nu sekvensen (3, 8, 5, 6, 2).

trasig gondol	ny gondol	möjlig gondolsekvens
1	6	(4, 5, 6, 2, 3)
4	7	(6, 2, 3, 7, 5)
7	8	(3, 8, 5, 6, 2)

En *ersättningssekvens* är en sekvens som består av numrerna på de gondoler som har gått sönder, i den ordning som de gick sönder. I det föregående exemplet så är ersättningssekvensen (1, 4, 7). Vi säger att en ersättningssekvens r *skapar* en gondolsekvens g om vi efter att gondolerna går sönder enligt ersättningssekvensen r får gondolsekvensen g.

Kontroll av gondolsekvens

För de första tre deluppgifterna så ska du undersöka om en viss given sekvens är en gondolsekvens. Se tabellen nedan för exempel på sekvenser som är och inte är gondolsekvenser. Du ska implementera funktionen `valid`.

- `valid(n, inputSeq)`
  - `n`: längden på inputsekvensen.
  - `inputSeq`: array av längd n; `inputSeq[i]` är element i för inputsekvensen, för $0 \leq i \leq n-1$.
  - Funktionen ska returnera 1 om inputsekvensen är en gondolsekvens, och annars ska den returnera 0.

Deluppgifter 1, 2, 3

deluppgift	poäng	n	`inputSeq`
1	5	$n \leq 100$	har varje tal mellan 1 och n exakt en gång
2	5	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq n$
3	10	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 250,000$

Exempel

deluppgift	`inputSeq`	returvärde	kommentar
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 kan ej förekomma precis före 5
1	(4, 3, 2, 1)	0	4 kan ej förekomma precis före 3
2	(1, 2, 3, 4, 5, 6, 5)	0	två gondoler med nummer 5
3	(2, 3, 4, 9, 6, 7, 1)	1	ersättningssekvens (5, 8)
3	(10, 4, 3, 11, 12)	0	4 kan ej förekomma precis före 3

Ersättningssekvens

För de nästa tre deluppgifterna så ska du konstruera en möjlig ersättningssekvens som skapar en given gondolsekvens. Om det finns flera ersättningssekvenser som skapar den givna gondolsekvensen så kan du välja vilken som helst. Du behöver implementera funktionen `replacement`.

- `replacement(n, gondolaSeq, replacementSeq)`
  - `n` är längden på gondolsekvensen.
  - `gondolaSeq`: array av längd n; `gondolaSeq` är garanterat en gondolsekvens och `gondolaSeq[i]` är gondol nummer i i sekvensen, för $0 \leq i \leq n-1$.
  - Funktionen ska returnera l, längden på ersättningsekvensen.

- `replacementSeq`: array som är tillräckligt stor för att lagra ersättningssekvensen; du anger din sekvens genom att placera gondol nummer i i ersättningssekvensen i `replacementSeq[i]`, för $0 \leq i \leq l - 1$.

Deluppgifter 4, 5, 6

deluppgift	poäng	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]` $\leq 250,000$

Exempel

deluppgift	`gondolaSeq`	returvärde	`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)

Räkna ersättningssekvenser

I de sista fyra deluppgifterna så ska du räkna antalet möjliga ersättningssekvenser som skapar en given gondolsekvens (vilken inte garanterat är en giltig gondolsekvens), modulo **1,000,000,009**. Du ska implementera funktionen `countReplacement`.

- `countReplacement(n, inputSeq)`
  - `n`: storleken på indata.
  - `inputSeq`: array av längd n; `inputSeq[i]` är element i i indatasekvensen, för $0 \leq i \leq n - 1$.
  - Om indatasekvensen är en gondolsekvens, så ska du beräkna antalet möjliga ersättningssekvenser (vilket kan vara extremt stort), *och returnera detta värde modulo **1,000,000,009***. Om indatasekvensen inte är en gondolsekvens så ska din funktion returnera 0. Om indatasekvensen är en gondolsekvens men inga gondoler har gått sönder, returnera 1.

Deluppgifter 7, 8, 9, 10

deluppgift	poäng	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, och minst $n - 3$ av gondolerna $1, \ldots, n$ gick inte sönder.
9	15	$n \leq 100,000$	$1 \leq$ `inputSeq[i]` $\leq 250,000$

deluppgift	poäng	n	inputSeq
10	10	$n \leq 100,000$	$1 \leq$ inputSeq[i] $\leq 1,000,000,000$

Exempel

deluppgift	inputSeq	returvärde	ersättningssekvens
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 är inte en gondolsekvens
10	(3, 4)	2	(1, 2) or (2, 1)

Implementationsdetaljer

Du ska submitta exakt en fil, vid namn gondola.c, gondola.cpp eller gondola.pas. Denna fil ska implementera subprogrammen som beskrivits ovan (alla tre, även om du inte tänker lösa alla deluppgifter), enligt de signaturer som följer nedan. Du ska också inkludera en header-fil gondola.h om du använder C/C++.

C/C++-program

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

Pascal-program

```
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;
```

Exempelrättare (sample grader)

Exempelrättare läser indata enligt följande format:

- rad 1: T, numret på deluppgiften som ditt program behandlar ($1 \leq T \leq 10$).
- rad 2: n, längden på indatasekvensen.
- rad 3: Om T är 4, 5, eller 6 så innehåller denna rad inputSeq[0], ..., inputSeq[n-1]. Annars innehåller denna rad gondolaSeq[0], ..., gondolaSeq[n-1].