



## Holiday

Jian-Jia planifică să-și petreacă următoarea vacanță în Taiwan. Pe durata vacanței, Jian-Jia va călători dintr-un oraș în altul și va vizita atracțiile orașelor.

Există  $n$  orașe în Taiwan, toate situate de-a lungul unei singure autostrăzi. Orașele sunt numerotate consecutiv de la 0 la  $n - 1$ . Pentru oricare oraș  $i$ , unde  $0 < i < n - 1$ , orașele adiacente sunt  $i - 1$  și  $i + 1$ . Orașul 0 este adiacent doar cu orașul 1, iar orașul  $n - 1$  este adiacent doar cu orașul  $n - 2$ .

Fiecare oraș conține un anumit număr de atracții. Jian-Jia are  $d$  zile de vacanță și intenționează să viziteze cât mai multe atracții posibile. Jian-Jia a ales deja orașul din care își va începe vacanța. În fiecare zi de vacanță, Jian-Jia poate călători în unul din orașele adiacente ori vizita toate atracțiile din orașul în care se află, dar nu poate face ambele. Jian-Jia *nu va vizita niciodată de mai multe ori atracțiile dintr-un oraș* chiar dacă vizitează același oraș de mai multe ori. Vă rugăm să-l ajutați pe Jian-Jia să-și planifice vacanța astfel încât să viziteze cât mai multe atracții posibile.

### Exemplu

Să presupunem că avem 5 orașe (listate în tabelul de mai jos), iar Jian-Jia are o vacanță de 7 zile pe care o începe din orașul 2. În prima zi, Jian-Jia va vizita cele 20 de atracții din orașul 2. În a doua zi Jian-Jia călătorește din orașul 2 în orașul 3, iar în a treia zi vizitează cele 30 de atracții din orașul 3. Apoi, Jian-Jia călătorește următoarele trei zile de la orașul 3 la orașul 0, și vizitează cele 10 atracții din orașul 0 în cea de-a șaptea zi. Numărul total de atracții pe care le va vizita Jian-Jia este  $20 + 30 + 10 = 60$ , care reprezintă numărul maxim de atracții pe care le poate vizita Jian-Jia în cele 7 zile de vacanță, în cazul în care primul oraș vizitat este 2.

oraș	număr de atracții
0	10
1	2
2	20
3	30
4	1

zi	acțiune
1	vizitează atracțiile din orașul 2
2	călătorește de la orașul 2 la orașul 3
3	vizitează atracțiile din orașul 3
4	călătorește de la orașul 3 la orașul 2
5	călătorește de la orașul 2 la orașul 1
6	călătorește de la orașul 1 la orașul 0
7	vizitează atracțiile din orașul 0

## Cerință

Trebuie să implementați funcția `findMaxAttraction` care calculează numărul maxim de atracții pe care Jian-Jia le poate vizita.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: numărul de orașe.
  - `start`: indicele primului oraș vizitat.
  - `d`: numărul de zile.
  - `attraction`: un tablou unidimensional de lungime `n`; `attraction[i]` este numărul de atracții din orașul `i`, pentru  $0 \leq i \leq n - 1$ .
  - Funcția trebuie să returneze numărul maxim de atracții pe care Jian-Jia le poate vizita.

## Subprobleme

În toate subproblemele  $0 \leq d \leq 2n + \lfloor n/2 \rfloor$  și numărul de atracții din fiecare oraș este nenegativ.

### Constrângeri adiționale:

subproblemă	puncte	$n$	numărul maxim de atracții dintr-un oraș	primul oraș vizitat
1	7	$2 \leq n \leq 20$	1,000,000,000	fără constrângeri
2	23	$2 \leq n \leq 100,000$	100	orașul 0
3	17	$2 \leq n \leq 3,000$	1,000,000,000	fără constrângeri
4	53	$2 \leq n \leq 100,000$	1,000,000,000	fără constrângeri

## Detalii de implementare

Trebuie să încărcați un singur fișier, cu numele `holiday.c`, `holiday.cpp` sau `holiday.pas`. Acest fișier va conține un subprogram care implementează subproblemele descrise mai sus folosind următorul antet. De asemenea trebuie să includeți headerul `holiday.h` pentru implementările C/C++.

Atenție: rezultatul poate fi un număr mare și tipul valorii returnate de funcția `findMaxAttraction` este un întreg pe 64-biți.

### pentru programele C/C++

```
long long int findMaxAttraction(int n, int start, int d,
int attraction[]);
```

## pentru programele Pascal

```
function findMaxAttraction(n, start, d : longint;  
attraction : array of longint) : int64;
```

### Grader-ul de pe computerul vostru

Grader-ul de pe computerul vostru citește datele de intrare în următorul format:

- linia 1: n, start, d.
- linia 2: attraction[0], ..., attraction[n-1].

Grader-ul de pe computerul vostru va afișa valoarea returnată de funcția `findMaxAttraction`.