Holiday

Jian-Jia planifică să-şi petreacă următoarea vacanţă în Taiwan. Pe durata vacanţei, Jian-Jia va călători dintr-un oraş în altul şi va vizita atracţiile oraşelor.

Există n oraşe în Taiwan, toate situate de-a lungul unei singure autostrăzi. Oraşele sunt numerotate consecutiv de la 0 la $n-1$. Pentru oricare oraş i, unde $0 < i < n-1$, oraşele adiacente sunt $i-1$ şi $i+1$. Oraşul 0 este adiacent doar cu oraşul 1, iar oraşul $n-1$ este adiacent doar cu oraşul $n-2$.

Fiecare oraş conţine un anumit număr de atracţii. Jian-Jia are d zile de vacanţă şi intenţionează să viziteze cat mai multe atracţii posibile. Jian-Jia a ales deja oraşul din care îşi va începe vacanţa. În fiecare zi de vacanţă, Jian-Jia poate călători în unul din oraşele adiacente ori vizita toate atracţiile din oraşul în care se află, dar nu poate face ambele. Jian-Jia *nu va vizita niciodată de mai multe ori atracţiile dintr-un oraş* chiar dacă vizitează acelaşi oraş de mai multe ori. Vă rugăm să-l ajutaţi pe Jian-Jia să-şi planifice vacanţa astfel încât să viziteze cât mai multe atracţii posibile.

Exemplu

Să presupunem că avem 5 oraşe (listate în tabelul de mai jos), iar Jian-Jia are o vacanţă de 7 zile pe care o începe din oraşul 2. În prima zi, Jian-Jia va vizita cele 20 de atracţii din oraşul 2. În a doua zi Jian-Jia călătoreşte din oraşul 2 în oraşul 3, iar în a treia zi vizitează cele 30 de atracţii din oraşul 3. Apoi, Jian-Jia călătoreşte următoarele trei zile de la oraşul 3 la oraşul 0, şi vizitează cele 10 atracţii din oraşul 0 în cea de-a şaptea zi. Numărul total de atracţii pe care le va vizita Jian-Jia este 20 + 30 + 10 = 60, care reprezintă numărul maxim de atracţiii pe care le poate vizita Jian-Jia în cele 7 zile de vacanţă, în cazul în care primul oraş vizitat este 2.

oraş	număr de atracţii
0	10
1	2
2	20
3	30
4	1

zi	acţiune
1	vizitează atracţiile din oraşul 2
2	călătoreşte de la oraşul 2 la oraşul 3
3	vizitează atracţiile din oraşul 3
4	călătoreşte de la oraşul 3 la oraşul 2
5	călătoreşte de la oraşul 2 la oraşul 1
6	călătoreşte de la oraşul 1 la oraşul 0
7	vizitează atracţiile din oraşul 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`.