Holiday

Jian-Jia plant zijn volgende vakantie in Taiwan. Tijdens deze vakantie reist JJ van stad naar stad en bezoekt daar attracties.

Er zijn n steden in Taiwan. Ze liggen aan één enkele lange snelweg. De steden zijn oplopend genummerd van 0 tot en met $n-1$. Voor stad i, $0 < i < n-1$, geldt dat steden $i-1$ en $i+1$ de buursteden zijn. Stad 0 heeft alleen stad 1 als buurstad. Stad $n-1$ heeft alleen stad $n-2$ als buurstad.

Elke stad heeft een aantal attracties. JJ heeft d dagen vakantie en wil zoveel mogelijk attracties bezoeken als mogelijk is. JJ heeft al besloten in welke stad hij zijn vakantie begint. Op elke dag van zijn vakantie kan JJ ofwel naar een buurstad reizen ofwel alle attracties in zijn huidige stad bezoeken, maar niet beide. JJ *bezoekt dezelfde attracties in een stad nooit twee keer*, zelfs niet als hij meerdere keren in de stad verblijft. Help JJ zijn vakantie te plannen opdat hij zoveel mogelijk verschillende attracties kan bezoeken.

Voorbeeld

Stel dat JJ 7 dagen vakantie heeft en dat er 5 steden zijn (zoals in de tabel hieronder). Hij begint zijn vakantie in stad 2. Op de eerste dag van zijn vakantie bezoekt hij de 20 attracties in stad 2. Op de tweede dag reist JJ van stad 2 naar stad 3, waar hij op de derde dag de 30 attracties bezoekt. De volgende drie dagen reist JJ van stad 3 naar stad 0, waar hij dan op de zevende dag de 10 attracties bezoekt. Het aantal attracties dat hij zo bezoekt is 20 + 30 + 10 = 60. Dit is het maximale aantal attracties dat JJ in 7 dagen bezoekt als hij begint in stad 2.

Stad	Aantal attracties
0	10
1	2
2	20
3	30
4	1

Dag	Wat doet JJ
1	Hij bezoekt de attracties in stad 2
2	Hij reist van stad 2 naar stad 3
3	Hij bezoekt de attracties in stad 3
4	Hij reist van stad 3 naar stad 2
5	Hij reist van stad 2 naar stad 1
6	Hij reist van stad 1 naar stad 0
7	Hij bezoekt de attracties in stad 0

Opdracht

Implementeer de functie `findMaxAttraction` die berekent wat het maximale aan attracties is dat JJ kan bezoeken.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: het aantal steden.
  - `start`: de index van de stad waarin JJ begint.
  - `d`: het aantal dagen.
  - `attraction`: array van lengte n; `attraction[i]` is het aantal attracties in stad i, waarbij $0 \leq i \leq n-1$.
  - De functie moet het maximale aantal attracties dat JJ kan bezoeken als resultaat leveren.

Subtasks

Voor alle subtasks geldt $0 \leq d \leq 2n + \lfloor n/2 \rfloor$. Het aantal attracties in elke stad is niet negatief.

Aanvullende voorwaarden:

subtask	punten	n	maximaal aantal attracties in een stad	stad waarin JJ begint
1	7	$2 \leq n \leq 20$	1 000 000 000	geen restricties
2	23	$2 \leq n \leq 100000$	100	stad 0
3	17	$2 \leq n \leq 3000$	1 000 000 000	geen restricties
4	53	$2 \leq n \leq 100000$	1 000 000 000	geen restricties

Implementatie details

Je moet precies een bestand inzenden, genaamd `holiday.c`, `holiday.cpp` of `holiday.pas`. Dit bestand moet het subprogramma zoals hierboven beschreven implementeren op basis van de hieronder beschreven signatures. Bij een C/C++ implementatie moet je ook de header file `holiday.h` includen.

Let op dat het antwoord groot kan zijn. Het type van het resultaat van `findMaxAttraction` is een 64-bit integer.

C/C++ programma

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

Pascal programma

```
function findMaxAttraction(n, start, d : longint;
```

```
  attraction : array of longint): int64;
```

Voorbeeld grader

De sample grader leest de invoer in het volgende formaat:

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

De sample grader drukt het resultaat van `findMaxAttraction` af.