Holiday

Jian-Jia plant zijn volgende vakantie in Taiwan. Tijdens deze vakantie reist Jian-Jia van stad naar stad en bezoekt hij attracties in de steden.

Er zijn n steden in Taiwan, allemaal gelegen langs een enkele snelweg. De steden zijn opeenvolgend genummerd van 0 tot $n-1$. Voor stad i, waarbij $0 < i < n-1$, zijn de buursteden $i-1$ and $i+1$. De enige buurstad van stad 0 is stad 1, en de enige buurstad van stad $n-1$ is stad $n-2$.

Elke stad bevat een aantal attracties. Jian-Jia heeft d dagen vakantie gepland en wil zoveel mogelijk attracties bezoeken. Hij heeft al een stad geselecteerd waar hij zijn vakantie wil beginnen. Elke vakantiedag kan Jian-Jia ofwel naar een buurstad reizen, ofwel alle attracties bezoeken van de stad waarin hij zich bevindt, maar niet allebei. Jian-Jia zal *de attracties van een stad nooit een tweede keer bezoeken*, zelfs als hij meermaals in de stad verblijft. Help Jian-Jia zijn vakantie te plannen zodat hij zoveel mogelijk verschillende attracties kan bezoeken.

Voorbeeld

Stel dat Jian-Jia 7 vakantiedagen heeft, dat er 5 steden zijn (zie de tabel hieronder), en dat hij start in stad 2. Op de eerste dag bezoekt Jian-Jia de 20 attracties in stad 2. Op de tweede dag reist hij naar stad 3. De derde dag bezoekt hij de 30 attracties in stad 3. Jian-Jia spendeert de volgende 3 dagen met reizen van stad 3 naar stad 0, en bezoekt dan de 10 attracties in stad 0 op de zevende dag. Het totale aantal attracties dat Jian-Jia bezoekt is 20 + 30 + 10 = 60, wat het maximale aantal attracties is dat Jian-Jia kan bezoeken in 7 dagen als hij start in stad 2.

stad	aantal attracties
0	10
1	2
2	20
3	30
4	1

dag	actie
1	bezoek attracties in stad 2
2	reis van stad 2 naar stad 3
3	bezoek attracties in stad 3
4	reis van stad 3 naar stad 2
5	reis van stad 2 naar stad 1
6	reis van stad 1 naar stad 0
7	bezoek attracties in stad 0

Taak

Implementeer een functie `findMaxAttraction` die het maximale aantal attracties berekent dat Jian-Jia kan bezoeken.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: het aantal steden.
  - `start`: de index van de stad waarin gestart wordt.
  - `d`: het aantal dagen.
  - `attraction`: array van lengte n; `attraction[i]` is het aantal attracties in stad i, voor $0 \leq i \leq n-1$.
  - De functie moet het maximale aantal attracties teruggeven dat Jian-Jia kan bezoeken.

Subtaken

In alle subtaken geldt: $0 \leq d \leq 2n + \lfloor n/2 \rfloor$, en het aantal attracties in een stad is nooit negatief.

Bijkomende beperkingen:

subtaak	punten	n	max. aantal attracties in een stad	eerste stad
1	7	$2 \leq n \leq 20$	1,000,000,000	geen beperkingen
2	23	$2 \leq n \leq 100,000$	100	stad 0
3	17	$2 \leq n \leq 3,000$	1,000,000,000	geen beperkingen
4	53	$2 \leq n \leq 100,000$	1,000,000,000	geen beperkingen

Implementatiedetails

Je moet exact één bestand indienen, genaamd `holiday.c`, `holiday.cpp` of `holiday.pas`. Dit bestand moet de hierboven beschreven subroutine implementeren volgens de volgende declaraties. Bij C/C++ implementatie moet je ook een header-file `holiday.h` "includeren".

Let op: het resultaat kan groot zijn, en het return-type 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;
```

Voorbeeldgrader

De voorbeeldgrader leest input in het volgende formaat:

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

De voorbeeldgrader zal de return-waarde van `findMaxAttraction` weergeven.