Holiday

Jian-Jia planerar sin nästa semester i Taiwan. Under denna semester så flyttar Jian-Jia från stad till stad och besöker attraktioner i staden.

Det finns n städer i Taiwan, alla ligger på samma väg. Städerna är numrerade i tur och ordning från 0 till $n-1$. För stad i, där $0 < i < n-1$, så gäller det att grannstäder är $i-1$ och $i+1$. Stad 0 är endast granne med stad 1, stad $n-1$ är endast granne med stad $n-2$.

Varje stad innehåller ett antal attraktioner. Jian-Jia ska semestra i d dagar och planerar att besöka så många attraktioner som möjligt. Jian-Jia har redan valt en stad som hen tänker påbörja semestern i. Under varje dag av semestern så kan Jian-Jia välja att antingen röra sig till en närliggande stad, eller besöka alla attraktioner i staden hen befinner sig i, men inte båda alternativen. Jian-Jia kommer *aldrig att besöka attraktionerna i samma stad flera gånger*, även om hen besöker samma stad flera gånger. Hjälp Jian-Jia planera sin semester så att hen besöker så många olika attraktioner som möjligt.

Exempel

Anta att Jian-Jia har 7 dagars semester, det finns 5 städer (listade i tabellen nedan) och hen börjar i stad 2. Första dagen så besöker hen de 20 attraktionerna i stad 2. Andra dagen flyttar hen sig från stad 2 till stad 3, och på den tredje dagen så besöker hen de 30 attraktionerna i stad 3. Jian-Jia spenderar sedan de kommande tre dagarna med att förflytta sig från stad 3 till stad 0, och besöker sedan de 10 attraktionerna i stad 0 på den sjunde dagen. Det totala antalet attraktioner som Jian-Jia besöker är 20 + 30 + 10 = 60, vilket är det maximala antalet attraktioner hen kan besöka när hen semestrar i 7 dagar och börjar i stad 2.

stad	antal attraktioner
0	10
1	2
2	20
3	30
4	1

stad	val
1	besöker attraktionerna i stad 2
2	förflyttar sig från stad 2 till stad 3
3	besöker attraktionerna i stad 3
4	förflyttar sig från stad 3 till stad 2
5	förflyttar sig från stad 2 till stad 1
6	förflyttar sig från stad 1 till stad 0
7	besöker attraktionerna i stad 0

Uppgift

Skriv en funktion `findMaxAttraction` som beräknar det maximala antalet attraktioner Jian-Jia kan besöka.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: antalet städer.
  - `start`: indexet för startstaden.
  - `d`: antalet dagar.
  - `attraction`: array av längd n; `attraction[i]` är antalet attraktioner i stad i, for $0 \leq i \leq n-1$.
  - Funktionen ska returnera det maximala antalet attraktioner som Jian-Jia kan besöka.

Deluppgifter

För alla deluppgifter så gäller det att $0 \leq d \leq 2n + \lfloor n/2 \rfloor$ samt att antalet attraktioner i varje stad är ickenegativt.

Ytterligare begränsningar:

deluppgift	poäng	n	maximalt antal attraktioner per stad	startstad
1	7	$2 \leq n \leq 20$	1,000,000,000	inga gränser
2	23	$2 \leq n \leq 100,000$	100	stad 0
3	17	$2 \leq n \leq 3,000$	1,000,000,000	inga gränser
4	53	$2 \leq n \leq 100,000$	1,000,000,000	inga gränser

Implementationsdetaljer

Du ska skicka in exakt en fil, med namn `holiday.c`, `holiday.cpp` eller `holiday.pas`. Denna fil ska implementera delprogrammet som beskrivits ovan och samtidigt har signaturer som följer nedan. Du ska också inkludera en header-fil `holiday.h` om du använder C/C++.

Notera att svaret kan vara jättestort, och att returtypen för `findMaxAttraction` är ett 64-bitars heltal.

C/C++-program

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

Pascal-program

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

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

Exempelrättare (sample grader)

Exempelrättaren läser indata med följande format:

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

Exempelrättaren kommer att skriva ut returvärdet av `findMaxAttraction`.