# Holiday

Jian-Jia sedang merencanakan liburan berikutnya di Taiwan. Selama liburannya, Jian-Jia berpindah dari satu kota ke kota lainnya, dan menyaksikan hal-hal menarik (*attractions*) di kota tersebut.

Ada $n$ kota di Taiwan, semua kota lokasinya pada satu jalan tol. Kota-kota tersebut dinomori secara berurutan dari 0 sampai dengan $n-1$. Untuk kota $i$, dengan $0 < i < n-1$, kota-kota bertetangga langsung adalah kota $i-1$ dan kota $i+1$. Satu-satunya kota yang bertetangga dengan kota 0 adalah kota 1, dan satu-satunya kota yang bertetangga dengan kota $n-1$ adalah kota $n-2$.

Pada setiap kota, ada beberapa hal yang menarik. Jian-Jia memiliki $d$ hari libur dan berencana mengunjungi hal menarik sebanyak mungkin. Jian-Jia sudah memilih kota di mana ia memulai liburannya. Setiap hari dalam liburannya, Jian-Jia dapat berpindah dari satu kota ke kota yang bertetangga lainnya, atau mengunjungi hal-hal menarik di kota di mana ia berada, tapi tak mungkin melakukan kedua hal tersebut. Jian-Jia tidak akan *pernah mengunjungi hal menarik di kota yang sama dua kali* walaupun ia berada pada kota tersebut lebih dari satu kali. Bantulah Jian-Jia merencanakan liburannya sehingga ia dapat mengunjungi sebanyak mungkin hal menarik.

## Example

Misalnya Jian-Jia mempunyai 7 hari libur, ada 5 kota (seperti ditunjukkan pada tabel di bawah), dan ia mulai dari kota 2. Pada hari pertama, Jian-Jia mengunjungi 20 hal-hal menarik di kota 2. Pada hari kedua, Jian-Jia berpindah dari kota 2 ke kota 3, dan pada hari ketiga ia mengunjungi 30 hal-hal menarik di kota 3. Kemudian, Jian-jia menghabiskan tiga hari berikutnya untuk berpindah secara berturut-turut dari kota 3 ke kota 0, dan mengunjungi 10 hal-hal menarik di kota 0 pada hari ketujuh. Banyaknya hal-hal menarik yang dikunjungi oleh Jian-Jia adalah 20 + 30 + 10 = 60, yaitu banyaknya maksimal hal-hal menarik yang dapat dilakukannya dalam 7 hari jika ia mulai berangkat dari kota 2.

| city | number of attractions |
|---|---|
| 0 | 10 |
| 1 | 2 |
| 2 | 20 |
| 3 | 30 |
| 4 | 1 |

| day | action |
|---|---|
| 1 | visit the attractions in city 2 |
| 2 | move from city 2 to city 3 |
| 3 | visit the attractions in city 3 |
| 4 | move from city 3 to city 2 |
| 5 | move from city 2 to city 1 |
| 6 | move from city 1 to city 0 |

| day | action |
|---|---|
| 7 | visit the attractions in city 0 |

# Task

Anda diminta untuk mengimplementasi sebuah function `findMaxAttraction` yang menghitung maksimal hal-hal menarik yang dapat dikunjungi Jian-Jia.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: the number of cities.
  - `start`: the index of the starting city.
  - `d`: the number of days.
  - `attraction`: array of length $n$; `attraction[i]` is the number of attractions in city $i$, for $0 \leq i \leq n-1$.
  - The function should return the maximum number of attractions Jian-Jia can visit.

# Subtasks

In all subtasks $0 \leq d \leq 2n + \lfloor n/2 \rfloor$.

### Additional constraints:

| subtask | points | $n$ | maximum number of attractions in a city | starting city |
|---|---|---|---|---|
| 1 | 7 | $2 \leq n \leq 20$ | 1,000,000,000 | no constraints |
| 2 | 23 | $2 \leq n \leq 100,000$ | 100 | city 0 |
| 3 | 17 | $2 \leq n \leq 3,000$ | 1,000,000,000 | no constraints |
| 4 | 53 | $2 \leq n \leq 100,000$ | 1,000,000,000 | no constraints |

# Implementation details

Anda harus mensubmisi sebuah file `holiday.c`, `holiday.cpp` or `holiday.pas`. File ini adalah implementasi dari subprogram yang dijelaskan di atas dengan *signature* sebagai berikut. Anda juga perlu melakukan *include* sebuah header file `holiday.h` untuk implementasi dalam C/C++ .

Perhatikanlah bahwa hasilnya mungkin besar, dan tipe dari nilai yang di *return* oleh `findMaxAttraction` adalah 64-bit integer.

### 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;
```

## Sample grader

The sample grader reads the input in the following format:

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

The sample grader will print the return value of `findMaxAttraction`.