Holiday

Jian-Jia vuole pianificare la sua prossima vacanza a Taiwan. Durante la vacanza, Jian-Jia si sposta di città in città e visita le attrazioni lì presenti.

Vi sono n città a Taiwan, tutte collocate lungo una singola autostrada. Le città sono numerate da 0 a $n-1$. Ogni città i, dove $0 < i < n-1$, è adiacente alle città $i-1$ e $i+1$. L'unica città adiacente alla città 0 è la città 1, e l'unica città adiacente alla città $n-1$ è $n-2$.

Ciascuna città contiene delle attrazioni. Jian-Jia ha d giorni di vacanza a sua disposizione e vuole visitare il maggior numero possibile di attrazioni. Jian-Jia ha già deciso la città da cui iniziare la sua vacanza. Ogni giorno della sua vacanza, Jian-Jia può muoversi verso una città adiacente, oppure può visitare tutte le attrazioni della città in cui si trova (ma non può scegliere entrambe le opzioni). Jian-Jia *non visita mai due volte le attrazioni della stessa città* anche se sceglie di stare nella stessa città più volte. Aiuta Jian-Jia a pianificare la sua vacanza in modo che possa visitare il maggior numero possibile di attrazioni.

Esempio

Supponi che Jian-Jia abbia 7 giorni di vacanza, ci siano 5 città (elencate nella tabella seguente), e che lui parta dalla città 2. Il primo giorno Jian-Jia visita le 20 attrazioni della città 2. Il secondo giorno, Jian-Jia si sposta dalla città 2 alla città 3, e il terzo visita le 30 attrazioni della città 3. Jian-Jia quindi trascorre i tre giorni successivi spostandosi dalla città 3 alla città 0, e visita le 10 attrazioni nella città 0 il settimo giorno. Il numero totale di attrazioni così visitate da Jian-Jia è 20 + 30 + 10 = 60, che è il massimo numero che può visitare in 7 giorni partendo dalla città 2.

città	numero di attrazioni
0	10
1	2
2	20
3	30
4	1

giorno	scelta
1	visita le attrazioni della città 2
2	spostati dalla città 2 alla città 3
3	visita le attrazioni della città 3
4	spostati dalla città 3 alla città 2
5	spostati dalla città 2 alla città 1
6	spostati dalla città 1 alla città 0
7	visita le attrazioni della città 0

Descrizione del problema

Devi implementare la funzione `findMaxAttraction` che calcola il massimo numero di attrazioni che Jian-Jia può visitare.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: numero di città.
  - `start`: indice della città di partenza.
  - `d`: numero di giorni.
  - `attraction`: array di lunghezza n; `attraction[i]` è il numero di attrazioni della città i, per $0 \leq i \leq n-1$.
  - La funzione deve restituire il massimo numero di attrazioni che Jian-Jia può visitare.

Subtasks

In tutti i subtask vale $0 \leq d \leq 2n + \lfloor n/2 \rfloor$, e il numero di attrazioni in ciascuna città è non-negativo.

Ulteriori vincoli:

subtask	punti	n	massimo numero di attrazioni per città	città di partenza
1	7	$2 \leq n \leq 20$	1 000 000 000	nessun vincolo
2	23	$2 \leq n \leq 100\,000$	100	città 0
3	17	$2 \leq n \leq 3\,000$	1 000 000 000	nessun vincolo
4	53	$2 \leq n \leq 100\,000$	1 000 000 000	nessun vincolo

Dettagli di implementazione

Devi sottoporre esattamente un file, chiamato `holiday.c`, `holiday.cpp` o `holiday.pas`. Questo file deve implementare la funzione descritta sopra, usando l'intestazione seguente. In C/C++, devi anche includere il file header `holiday.h`.
Notare che il risultato potrebbe essere molto grande, e che il tipo restituito da `findMaxAttraction` è un intero a 64-bit.

Linguaggio C/C++

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

Linguaggio Pascal

```
function findMaxAttraction(n, start, d : longint;
attraction : array of longint): int64;
```

Grader di esempio

Il grader di esempio legge l'input secondo il formato seguente:

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

Il grader di esempio stamperà il valore restituito da `findMaxAttraction`.