Urlaub

Jian-Jia plant seinen nächsten Urlaub in Taiwan. Während seines Urlaubs geht Jian-Jia von Stadt zu Stadt und besucht Sehenswürdigkeiten in den Städten.

Es gibt n Städte in Taiwan, die sich alle entlang einer einzigen Autobahn befinden. Die Städte sind aufeinanderfolgend nummeriert von 0 bis $n-1$. Die Stadt i, wobei $0 < i < n-1$, ist benachbart zu den Städten $i-1$ und $i+1$. Die einzige Stadt, die zu Stadt 0 benachbart ist, ist die Stadt 1. Die einzige Stadt, die zu Stadt $n-1$ benachbart ist, ist die Stadt $n-2$.

Jede Stadt enthält eine gewisse Anzahl an Sehenswürdigkeiten. Jian-Jia hat d Tage Urlaub und plant, so viele Sehenswürdigkeiten wie möglich zu besichtigen. Jian-Jia hat bereits eine Stadt gewählt, in welcher er seinen Urlaub beginnen wird. An jedem Urlaubstag kann Jian-Jia sich entweder zu einer benachbarten Stadt bewegen oder alle Sehenswürdigkeiten der Stadt, in der er sich gerade befindet, besichtigen. Aber nicht beides. Jian-Jia wird *nie die Sehenswürdigkeiten derselben Stadt zweimal besichtigen*, auch nicht wenn er sich mehrmals in der gleichen Stadt aufhält. Bitte hilf Jian-Jia seinen Urlaub so zu planen, dass er so viele Sehenswürdigkeiten wie möglich besichtigen kann.

Beispiel

Nimm an, dass Jian-Jia 7 Tage Ferien hat, es 5 Städte gibt (aufgelistet in der Tabelle unten), und er in Stadt 2 startet. Am ersten Tag besichtigt Jian-Jia 20 Sehenswürdigkeiten in Stadt 2. Am zweiten Tag reist Jian-Jia von Stadt 2 nach Stadt 3 und am dritten Tag besucht er die 30 Sehenswürdigkeiten in Stadt 3. Jian-Jia verbringt dann die nächsten drei Tage damit, von Stadt 3 zu Stadt 0 zu reisen und besichtigt die 10 Sehenswürdigkeiten in Stadt 0 am siebten Tag. Die Gesamtzahl an Sehenswürdigkeiten, die Jian-Jia besichtigt, ist 20 + 30 + 10 = 60. Dies ist die maximale Anzahl an Sehenswürdigkeiten, welche Jian-Jia in 7 Tagen besichtigen kann, wenn er in Stadt 2 startet.

Stadt	Anzahl an Sehenswürdigkeiten
0	10
1	2
2	20
3	30
4	1

Tag	Handlung
1	Sehenswürdigkeiten in Stadt 2 besuchen
2	von Stadt 2 nach Stadt 3 reisen
3	Sehenswürdigkeiten in Stadt 3 besuchen
4	von Stadt 3 nach Stadt 2 reisen
5	von Stadt 2 nach Stadt 1 reisen
6	von Stadt 1 nach Stadt 0 reisen

Tag	Handlung
7	Sehenswürdigkeiten in Stadt 0 besuchen

Task

Bitte implementiere eine Funktion `findMaxAttraction`, welche die maximale Anzahl an Sehenswürdigkeiten berechnet, die Jian-Jia besuchen kann.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: die Anzahl Städte
  - `start`: der Index der Start-Stadt
  - `d`: die Anzahl Urlaubstage
  - `attraction`: ein Array der Länge n; `attraction[i]` ist die Anzahl an Sehenswürdigkeiten in Stadt i, für $0 \leq i \leq n-1$.
  - Die Funktion soll die maximale Anzahl an Sehenswürdigkeiten zurückgeben, die Jian-Jia besuchen kann.

Subtasks

In allen Subtasks gilt $0 \leq d \leq 2n + \lfloor n/2 \rfloor$ und die Anzahl Sehenswürdigkeiten in jeder Stadt ist nicht negativ.

Zusätzliche Limits:

Subtask	Punkte	n	maximale Anzahl Sehenswürdigkeiten in einer Stadt	Start-Stadt
1	7	$2 \leq n \leq 20$	1 000 000 000	keine Einschränkung
2	23	$2 \leq n \leq 100\,000$	100	Stadt 0
3	17	$2 \leq n \leq 3\,000$	1 000 000 000	keine Einschränkung
4	53	$2 \leq n \leq 100\,000$	1 000 000 000	keine Einschränkung

Implementierungsdetails

Du musst exakt eine Datei, genannt `holiday.c`, `holiday.cpp` oder `holiday.pas`, einschicken. Diese Datei muss die oben beschriebene Funktion mit den folgenden Signaturen implementieren. Für C/C++ musst du ausserdem die Header-Datei `holiday.h` einbinden.

Beachte, dass das Resultat gross sein kann und der Rückgabewert von `findMaxAttraction` eine 64-Bit Ganzzahl ist.

Programme in C/C++

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

Programme in Pascal

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

Sample-Grader

Der Sample-Grader liest die Eingabe in folgendem Format ein:

- Erste Zeile: `n`, `start`, `d`.
- Zweite Zeile: `attraction[0]`, ..., `attraction[n-1]`.

Der Sample-Grader gibt den Rückgabewert von `findMaxAttraction` aus.