Holiday

Jian-Jia veut planifier ses prochaines vacances à Taiwan. Durant son séjour, Jian-Jia se déplace d'une ville à une autre et visite les attractions de ces villes.

Il y a à Taiwan n villes, toutes situées le long d'une seule route. Les villes sont numérotées de 0 à $n-1$. Pour la ville i, pour $0 \leq i \leq n-1$, les villes adjacentes sont $i-1$ et $i+1$. L'unique ville adjacente à la ville 0 est la ville 1 et l'unique ville adjacente à la ville $n-1$ est la ville $n-2$.

Chaque ville contient un nombre donné d'attractions. Jian-Jia a un certain nombre d de jours de vacances et projette de visiter autant d'attractions que possible. Il a déjà choisi une ville dans laquelle il commence son séjour. Pour chaque jour de ses vacances, il peut soit se déplacer vers une ville adjacente soit visiter toutes les attractions de la ville où il séjourne, mais pas les deux. Jian-Jia ne *visitera jamais les attractions d'une même ville deux fois* même s'il y séjourne plusieurs fois. Aidez Jian-Jia à planifier son séjour de façon à ce qu'il visite le plus d'attractions différentes possibles.

Exemple

Supposons que le séjour de Jian-Jia dure 7 jours, qu'il y a 5 villes (listées dans le tableau ci-après), et qu'il commence à la ville 2. Lors du 1er jour, il visite les 20 attractions de la ville 2. Le 2ème jour, il passe de la ville 2 à la ville 3, et le 3ème jour il visite les 30 attractions de la ville 3. Jian-Jia passe les trois jours suivants à se déplacer de la ville 3 à la ville 0 et il visite les 10 attractions de la ville 0 le 7ème jour. Le nombre total d'attractions que Jian-Jia a visité est 20 + 30 + 10 = 60, c'est le nombre maximal d'attractions que Jian-Jia peut visiter lors des 7 jours en partant de la ville 2.

ville	nombre d'attractions
0	10
1	2
2	20
3	30
4	1

jour	action
1	visite les attractions de la ville 2
2	se déplace de la ville 2 à la ville 3
3	visite les attractions de la ville 3
4	se déplace de la ville 3 à la ville 2
5	se déplace de la ville 2 à la ville 1
6	se déplace de la ville 1 à la ville 0
7	visite les attractions de la ville 0

Tâche

Vous devez implémenter une fonction `findMaxAttraction` qui calcule le nombre maximal d'attractions que Jian-Jia peut visiter.

- `findMaxAttraction(n, start, d, attraction)`
  - `n` : le nombre de villes.
  - `start` : la ville de départ.
  - `d` : le nombre de jours.
  - `attraction` : tableau de taille n; `attraction[i]` est le nombre d'attractions dans la ville i, pour $0 \leq i \leq n-1$.
  - La fonction doit retourner le nombre maximal d'attractions que Jian-Jia peut visiter.

Sous-tâches

Dans l'ensemble des sous-tâches, $0 \leq d \leq 2n + \lfloor n/2 \rfloor$ et le nombre d'attractions de chaque ville est positif ou nul.

Contraintes additionnelles :

sous-tâche	points	n	nombre maximal d'attractions par ville	ville de départ
1	7	$2 \leq n \leq 20$	1.000.000.000	aucune contrainte
2	23	$2 \leq n \leq 100.000$	100	ville 0
3	17	$2 \leq n \leq 3.000$	1.000.000.000	aucune contrainte
4	53	$2 \leq n \leq 100.000$	1.000.000.000	aucune contrainte

Détails d'implémentation

Vous devez soumettre un seul fichier, nommé `holiday.c`, `holiday.cpp` ou `holiday.pas`. Ce fichier doit implémenter la fonction décrite précédemment en utilisant une des signatures suivantes. Vous devez inclure (#include) l'en-tête `holiday.h` pour les programmes en C/C++.

Notez que le résultat peut atteindre une grande valeur, et que le type de retour de `findMaxAttraction` est un entier 64 bits.

Programme C/C++

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

Programme Pascal

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

Evaluateur

L'évaluateur lit l'entrée en suivant ce format

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

L'évaluateur affichera la valeur de retour de `findMaxAttraction`.