Holiday (Vacaciones)

Jian-Jia planea sus siguientes vacaciones en Taiwan. Durante ellas, Jian-Jia se traslada de una ciudad a otra y visita las atracciones de cada una de las ciudades.

Existen n ciudades en Taiwan, todas están conectadas por una única carretera. Las ciudades se numeran de forma consecutiva desde 0 hasta $n-1$. Para la ciudad i, donde $0 < i < n-1$, las ciudades adyacentes son $i-1$ e $i+1$. La única ciudad adyacente a la 0 es la ciudad 1, y la única ciudad adyacente a la $n-1$ es la ciudad $n-2$.

Cada ciudad tiene una número de atracciones. Jian-Jia tiene d días de vacaciones y planea visitar tantas atracciones como le sea posible. Jian-Jia ha seleccionado una ciudad en donde va a iniciar sus vacaciones. Durante cada día, Jian-Jia puede trasladarse a una ciudad adyacente o visitar las atracciones de la ciudad en la que se encuentra, pero no puede hacer ambas. Jian-Jia *nunca visita dos veces las atracciones de una misma ciudad* aun si se queda múltiples días en una misma ciudad. Ayuda a Jian-Jia a planear sus vacaciones de modo que visite tantas atracciones distintas como le sea posible.

Ejemplo

Supón que Jian-Jia tiene 7 días de vacaciones, hay 5 ciudades (listadas en la tabla debajo) e inicia en la ciudad 2. En el primer día Jian-Jia visita las 20 atracciones de la ciudad 2, en el segundo día Jian-Jia se traslada de la ciudad 2 a la ciudad 3, en el tercer día visita las 30 atracciones de la ciudad 3, luego Jian-Jia utiliza los siguientes 3 días para trasladarse de la ciudad 3 a la ciudad 0, finalmente en el séptimo día visita las 10 atracciones de la ciudad 0. El número total de atracciones que Jian-Jia visitó es 20 + 30 + 10 = 60, que es el máximo número de atracciones que Jian-Jia puede visitar en 7 días iniciando desde la ciudad 2.

ciudad	número de atracciones
0	10
1	2
2	20
3	30
4	1

día	acción
1	visita las atracciones en la ciudad 2
2	traslado de la ciudad 2 a la ciudad 3
3	visita las atracciones en la ciudad 3
4	traslado de la ciudad 3 a la ciudad 2
5	traslado de la ciudad 2 a la ciudad 1
6	traslado de la ciudad 1 a la ciudad 0
7	visita las atracciones en la ciudad 0

Problema

Implementa la función `findMaxAttraction` que calcule el máximo número de atracciones que Jian-Jia puede visitar.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: el número de ciudades.
  - `start`: el índice de la ciudad inicial.
  - `d`: el número de días.
  - `attraction`: arreglo de longitud n; `attraction[i]` es el número de atracciones en la ciudad i, para $0 \le i \le n-1$.
  - La función debe devolver el máximo número de atracciones que Jian-Jia puede visitar.

Subproblemas

En todos los subproblemas $0 \le d \le 2n + \lfloor n/2 \rfloor$ y el número de atracciones en cada ciudad es no negativo.

Límites adicionales:

subproblema	puntos	n	máximo número de atracciones en una ciudad (t)	ciudad de inicio
1	7	$2 \le n \le 20$	$0 \le t \le 1,000,000,000$	cualquiera
2	23	$2 \le n \le 100,000$	$0 \le t \le 100$	ciudad 0
3	17	$2 \le n \le 3,000$	$0 \le t \le 1,000,000,000$	cualquiera
4	53	$2 \le n \le 100,000$	$0 \le t \le 1,000,000,000$	cualquiera

Detalles de implementación

Debes enviar exactamente un archivo llamado `holiday.c`, `holiday.cpp` o `holiday.pas`. Este archivo debe implementar la función descrita arriba de acuerdo a los siguientes prototipos. También debes incluir el archivo de cabecera `holiday.h` para el caso de las implementaciones en C/C++.

Nota que el resultado puede ser muy grande, el tipo de dato que devuelve `findMaxAttraction` debe ser un entero de 64-bit.

Programa en C/C++

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

Programa en Pascal

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

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada de acuerdo al siguiente formato:

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

El evaluador de ejemplo imprimira el valor devuelto por `findMaxAttraction`.