Holiday

Jian-Jia está planeando sus próximas vacaciones en Taiwán. Durante sus vacaciones, Jian-Jia puede moverse de ciudad en ciudad y visitar atracciones en cada una de las ciudades.

Hay n ciudades en Taiwán, todas dispuestas a lo largo de una sola línea y están numeradas consecutivamente de 0 a $n-1$. Para la ciudad i, donde $0 < i < n-1$, las ciudades adyacentes son la $i-1$ y la $i+1$. La única ciudad adyacente a la ciudad 0 es la ciudad 1, y la única ciudad adyacente a la ciudad $n-1$ es la ciudad $n-2$.

Cada ciudad contiene cierto número de atracciones. Jian-Jia tiene d días de vacaciones y planea visitar tantas atracciones como sea posible. Jian-Jia ya ha seleccionado una ciudad donde partir sus vacaciones. En cada día de sus vacaciones Jian-Jia puede moverse a una ciudad adyacente o visitar todas las atracciones de la ciudad en la que se encuentra, pero no ambas. Jian-Jia *nunca visitará las atracciones en una ciudad más de una vez* incluso si se queda en la ciudad más de una vez. Por favor ayuda a Jian-Jia a planear sus vacaciones de forma que pueda visitar la mayor cantidad de atracciones posible.

Ejemplo

Supón que Jian-Jia tiene 7 días de vacaciones, hay 5 ciudades (listadas en la tabla de abajo) y además comienza en la ciudad 2. En el primer día Jian-Jia visita las 20 atracciones en la ciudad 2. En el segundo día Jian-Jia se mueve desde la ciudad 2 a la ciudad 3, y en el tercer día visita las 30 atracciones en la ciudad 3. Jian-Jia gasta posteriormente los siguientes tres días moviéndose desde la ciudad 3 a la ciudad 0 y luego visita las 10 atracciones en la ciudad 0 en el séptimo día. El número total de atracciones que Jian-Jia visita es 20 + 30 + 10 = 60, que es el máximo número de atracciones que Jian-Jia puede visitar en 7 días cuando parte de la ciudad 2.

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

día	acción
1	visitar las atracciones en la ciudad 2
2	moverse de la ciudad 2 a la ciudad 3
3	visitar las atracciones en la ciudad 3
4	moverse de la ciudad 3 a la ciudad 2
5	moverse de la ciudad 2 a la ciudad 1
6	moverse de la ciudad 1 a la ciudad 0

día	acción
7	visitar las atracciones en la ciudad 2

Tarea

Por favor implementa la función `findMaxAttraction` que calcula la máxima cantidad de atracciones que puede visitar Jian-Jia.

- `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 tamaño n; `attraction[i]` contiene el número de atracciones en la ciudad i, para $0 \leq i \leq n-1$
- La función debe retornar el número máximo de atracciones que Jian-Jia puede visitar.

Subtareas

En todas las subtareas $0 \leq d \leq 2n + \lfloor n/2 \rfloor$, y el número de atracciones en cada ciudad es no negativo.

Restricciones adicionales:

subtarea	puntos	n	número máximo de atracciones en una ciudad	ciudad de inicio
1	7	$2 \leq n \leq 20$	$0 \leq t \leq 1,000,000,000$	sin restricciones
2	23	$2 \leq n \leq 100,000$	$0 \leq t \leq 100$	ciudad 0
3	17	$2 \leq n \leq 3,000$	$0 \leq t \leq 1,000,000,000$	sin restricciones
4	53	$2 \leq n \leq 100,000$	$0 \leq t \leq 1,000,000,000$	sin restricciones

Detalles de Implementación

Debes enviar exactamente un archivo, llamado `holiday.c`, `holiday.cpp` o `holiday.pas`. Este archivo debe contener la implementación del subprograma descrito anteriormente y usando la siguiente firma. Adicionalmente debes incluir el archivo `holiday.h` para C/C++.

Notar que el resultado puede ser muy largo y el tipo de retorno de `findMaxAttraction` es un entero de 64-bit.

C/C++

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

Pascal

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

Grader de ejemplo

El *grader* de ejemplo lee el input en el siguiente formato:

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

El *grader* de ejemplo imprimirá el valor retornado por `findMaxAttraction`.