



# Holiday

Jian-Jia está planificando sus próximas vacaciones en Taiwan. Durante sus vacaciones, Jian-Jia se desplaza de ciudad en ciudad y visita atractivos en las ciudades.

Hay  $n$  ciudades en Taiwan, todas ubicadas a lo largo de una sola autopista. Las ciudades están numeradas consecutivamente desde 0 hasta  $n - 1$ . Para la ciudad  $i$ , donde  $0 < i < n - 1$ , las ciudades adyacentes son  $i - 1$  y  $i + 1$ . 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 atractivos. Jian-Jia planea visitar tantos atractivos como le sea posible. Jian-Jia ya ha seleccionado la ciudad en la cual iniciar sus vacaciones. Cada día Jian-Jia puede, ya sea desplazarse a una ciudad adyacente, ya sea visitar todos los atractivos de la ciudad en la cual está, pero no ambos. Jian-Jia *nunca visitará los atractivos en una misma ciudad dos veces* aunque estuviera en la ciudad múltiples veces. Por favor ayuda a Jian-Jia a planear sus vacaciones de modo tal que visite tantos atractivos diferentes como sea posible.

## Ejemplo

Suponga que Jian-Jia tiene 7 días de vacaciones, hay 5 ciudades (listadas en la tabla abajo), y que él parte desde la ciudad 2. Durante el primer día Jian-Jia visita los 20 atractivos de la ciudad 2. En el segundo día Jian-Jia se desplaza desde la ciudad 2 hacia la ciudad 3, y en el tercer día visita los 30 atractivos en la ciudad 3. Jian-Jia entonces insume los tres siguientes días en desplazarse desde la ciudad 3 hacia la ciudad 0, y visita los 10 atractivos de la ciudad 1 en el sexto día. El número total de atractivos Jian-Jia visita es  $20 + 30 + 10 = 60$ , el cual es el máximo número de atractivos Jian-Jia puede visitar en 7 días cuando él inicia desde la ciudad 2.

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

día	acción
1	visita los atractivos en la ciudad 2
2	se desplaza desde la ciudad 2 hacia la ciudad 3
3	visita los atractivos en la ciudad 3
4	se desplaza desde la ciudad 3 hacia la ciudad 2
5	se desplaza desde la ciudad 2 hacia la ciudad 1
6	se desplaza desde la ciudad 1 hacia la ciudad 0

día	acción
7	visita los atractivos en la ciudad 0

## Tarea

Por favor implementa una función `findMaxAttraction` que compute el máximo número de atractivos Jian-Jia puede visitar.

- `findMaxAttraction start, d, attraction)`
  - `n`: el número de ciudades.
  - `start`: el índice de la ciudad de inicio.
  - `d`: el número de días.
  - `attraction`: array de largo `n`; `attraction[i]` es el número de atractivos en la ciudad `i`, para  $0 \leq i \leq n - 1$ .
  - La función debe devolver el máximo número de atractivos Jian-Jia puede visitar.

## Subtareas

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

### Restricciones adicionales:

subtarea	puntos	$n$	máximo número de atractivos en una ciudad	ciudad de inicio
1	7	$2 \leq n \leq 20$	1.000.000.000	sin restricciones
2	23	$2 \leq n \leq 100.000$	100	la ciudad 0
3	17	$2 \leq n \leq 3.000$	1.000.000.000	sin restricciones
4	53	$2 \leq n \leq 100.000$	1.000.000.000	sin restricciones

## Detalles de implementación

Debes enviar exactamente un archivo, llamado `holiday.c`, `holiday.cpp` o `holiday.pas`. Este archivo debiera implantar el subprograma descrito más arriba utilizando los siguientes encabezamientos. Necesitas incluir también un archivo cabeza `holiday.h` para la implantación en C/C++.

Observe que el resultado puede ser grande, y el tipo de retorno de `findMaxAttraction` es un entero de 64bits.

## programación en C/C++

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

## Programación en Pascal

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

## Sample grader

El sample grader (programa evaluador para prueba local) lee su entrada con el formato siguiente:

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

El sample grader imprimirá el valor de retorno de findMaxAtractivo.