Holiday

O Jian-Jia está a planear as suas próximas férias em Taiwan. Durante as suas férias, o Jian-Jia desloca-se de cidade para cidade e visita atrações nas cidades.

Há n cidades em Taiwan, todas localizadas numa única autoestrada. As cidades estão numeradas consecutivamente de 0 a $n-1$. Para a cidade i, onde $0 < i < n-1$, as cidades adjacentes são a $i-1$ e a $i+1$. A única cidade adjacente à cidade 0 é a cidade 1 e a única cidade adjacente à cidade $n-1$ é a cidade $n-2$.

Cada cidade contém um certo número de atrações. O Jian-Jia tem d dias de férias e planeia visitar o maior número possível de atrações. O Jian-Jia já selecionou uma cidade para começar as suas férias. Em cada dia das suas férias, o Jian-Jia pode ou mover-se para uma cidade adjacente ou visitar todas as atrações da cidade onde se encontra, mas não ambas as coisas. O Jian-Jia *nunca visita as atrações da mesma cidade duas vezes*, mesmo que fique na mesma cidade múltiplas vezes. Por favor ajuda o Jian-Jia a planear as suas férias de maneira a visitar o maior número possível de atrações.

Exemplo

Suponha que o Jian-Jia tem 7 dias de férias, que há 5 cidades (listadas na tabela abaixo) e que ele começa da cidade 2. No primeiro dia o Jian-Jia visita as 20 atrações da cidade 2. No segundo dia o Jian-Jia move-se da cidade 2 para a cidade 3 e no terceiro dia visita as 30 atrações da cidade 3. O Jian-Jia depois passa os próximos três dias a mover-se da cidade 3 para a cidade 0 e visita as 10 atrações da cidade 0 no sétimo dia. No total o Jian-Jia visita 20 + 30 + 10 = 60 atrações, que é o número máximo de atrações que o Jian-Jia consegue visitar em 7 dias quando ele começa da cidade 2.

cidade	número de atrações
0	10
1	2
2	20
3	30
4	1

dia	ação
1	visita as atrações da cidade 2
2	move-se da cidade 2 para a cidade 3
3	visita as atrações da cidade 3
4	move-se da cidade 3 para a cidade 2
5	move-se da cidade 2 para a cidade 1
6	move-se da cidade 1 para a cidade 0

dia	ação
7	visita as atrações da cidade 0

Tarefa

Por favor implementa a função `findMaxAttraction` que calcula o número máximo de atrações que o Jian-Jia pode visitar.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: o número de cidades.
  - `start`: o índice da cidade onde o Jian-Jia começa.
  - `d`: o número de dias.
  - `attraction`: vetor de tamanho n; `attraction[i]` é o número de atrações da cidade i, para $0 \leq i \leq n-1$.
  - A função deve retornar o número máximo de atrações que o Jian-Jia pode visitar.

Subtarefas

Em todas as subtarefas $0 \leq d \leq 2n + \lfloor n/2 \rfloor$, e o número de atrações em cada cidade é não negativo.

Restrições adicionais:

subtarefa	pontos	n	número máximo de atrações d cidade	cidade por onde o Jian-Jia começa
1	7	$2 \leq n \leq 20$	1,000,000,000	sem restrições
2	23	$2 \leq n \leq 100,000$	100	cidade 0
3	17	$2 \leq n \leq 3,000$	1,000,000,000	sem restrições
4	53	$2 \leq n \leq 100,000$	1,000,000,000	sem restrições

Detalhes de implementação

Tens de submeter exatamente um ficheiro chamado `holiday.c`, `holiday.cpp` ou `holiday.pas`. Este ficheiro (arquivo) deve implementar o subprograma descrito acima usando as seguintes assinaturas. Terás também de incluir o ficheiro `holiday.h` para a implementação em C/C++.

Nota que o resultado pode ser grande e o tipo de retorno da função `findMaxAttraction` é um inteiro de 64 bits.

Programa em C/C++

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

Programa em Pascal

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

Avaliador exemplo

O avaliador exemplo lê a entrada no seguinte formato:

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

O avaliador exemplo imprime o valor de retorno da função `findMaxAttraction`.