Blagdan

Janko radi plan za svoj idući godišnji odmor koji će provesti na Tajvanu. Za vrijeme godišnjeg odmora putovat će od grada do grada i obilaziti kulturne znamenitosti.

Na Tajvanu se nalazi n gradova povezanih jednom autocestom. Označeni su brojevima $0, 1, \ldots, n-1$. Gradu $i (0 < i < n-1)$ susjedni su gradovi $i-1$ i $i+1$. Logično, jedini grad susjedan gradu 0 je grad 1, a jedini grad susjedan gradu $n-1$ je grad $n-2$.

U svakom gradu nalazi se neki broj kulturnih znamenitosti, a naš Janko u svojih d dana odmora želi ih posjetiti što više. Janko se na početku prvog dana nalazi u nekom od gradova. Svaki dan godišnjeg odmora Janko može provesti putujući iz grada u njemu susjedni grad ili obilazeći sve znamenitosti grada u kojem se trenutno nalazi. Ne može raditi oboje pa mora dobro isplanirati put. Također, Janko *nikada neće obići znamenitosti istog grada više od jednom*. Pomozite Janku da obiđe što više *različitih* znamenitosti za vrijeme svog odmora.

Primjer

Neka Jankov godišnji traje 7 dana i neka postoji 5 gradova na Tajvanu. Janko u obilazak kreće iz grada 2. Prvog dana odmora posjetit će 20 znamenitosti u gradu 2. Drugog dana otputovat će iz grada 2 u njemu susjedan grad 3, a trećeg dana obići će 30 znamenitosti u njemu. Četvrti, peti i šesti dan odmora putovat će od grada 3 do grada 1. Sedmi dan posjetit će 10 znamenitosti u gradu 1. Ukupan broj znamenitosti koji je Janko obišao je $20 + 30 + 10 = 60$, što je ujedno i najviše što može u šest dana ako kreće iz grada 2.

grad	broj znamenitosti
0	10
1	2
2	20
3	30
4	1

dan	što radi Janko?
1	obilazi znamenitosti u gradu 2
2	putuje iz grada 2 u grad 3
3	obilazi znamenitosti u gradu 3
4	putuje iz grada 3 u grad 2
5	putuje iz grada 2 u grad 1
6	putuje iz grada 1 u grad 0
7	obilazi znamenitosti u gradu 0

Zadatak

Implementirajte funkciju `findMaxAttraction` koja računa najveći broj znamenitosti koje Janko može obići.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: broj gradova
  - `start`: redni broj početnog grada
  - `d`: broj dana
  - `attraction`: niz duljine n; `attraction[i]` je broj znamenitosti u gradu i za sve $i \in \{0, 1, \ldots, n-1\}$
  - funkcija mora vratiti najveći broj znamenitosti koje Janko može obići

Podzadaci

U svim podzadacima vrijedit će $0 \leq d \leq 2n + \lfloor n/2 \rfloor$ i broj atrakcija u svakom gradu bit će nenegativan cijeli broj.

Dodatna ograničenja:

podzadatak	broj bodova	n	najveći mogući broj znamenitosti u gradu (t)	početni grad
1	7	$2 \leq n \leq 20$	$0 \leq t \leq 1,000,000,000$	bilo koji
2	23	$2 \leq n \leq 100,000$	$0 \leq t \leq 100$	grad 0
3	17	$2 \leq n \leq 3,000$	$0 \leq t \leq 1,000,000,000$	bilo koji
4	53	$2 \leq n \leq 100,000$	$0 \leq t \leq 1,000,000,000$	bilo koji

Implementacijski detalji

Morate *submitati* točno jednu datoteku, `holiday.c`, `holiday.cpp` ili `holiday.pas`. U ovoj datoteci mora biti implementiran gore opisani potprogram sa dolje navedenim prototipovima. Također, morate *includeati header file* `holiday.h` za C/C++.

C/C++ program

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

Paskal program

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

Sample grader

Sample grader prima ulaz sljedećeg oblika:

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

Sample grader će ispisati povratnu vrijednost funkcije `findMaxAttraction`.