Puhkus

Jian-Jia planeerib oma järgmist puhkust Taiwanil. Puhkuse ajal liigub Jian-Jia linnast linna ja külastab linnades vaatamisväärsusi.

Taiwanil on n linna, mis kõik asuvad sama tee ääres. Linnadel on järjestikused numbrid 0 kuni $n-1$. Linna i (kus $0 < i < n-1$) naaberlinnad on $i-1$ ja $i+1$. Linna 0 ainus naaberlinn on linn 1 ja linna $n-1$ ainus naaberlinn on $n-2$.

Igas linnas on mingi hulk vaatamisväärsusi. Jian-Jial on d puhkusepäeva ning ta tahab näha nii palju vaatamisväärsusi kui võimalik. Jian-Jia on juba valinud linna, kust ta oma puhkust alustab. Igal päeval saab Jian-Jia kas liikuda naaberlinna või siis vaadata kõiki vaatamisväärsusi linnas, kus ta viibib, aga mitte mõlemat. Jian-Jia *ei vaata kunagi sama linna vaatamisväärsusi kaks korda*, isegi juhul, kui ta peatub selles linnas mitu korda. Aita Jian-Jial oma puhkus selliselt planeerida, et ta näeks nii paljusid vaatamisväärsusi kui võimalik.

Näide

Oletame, et Jian-Jial on 7 päeva puhkust, tee ääres on 5 linna (vt tabelit) ja ta alustab linnast 2. Esimesel päeval külastab Jian-Jia 20 vaatamisväärsust linnas 2. Teisel päeval liigub Jian-Jia linnast 2 linna 3 ja kolmandal päeval külastab ta 30 vaatamisväärsust linnas 3. Seejärel kulutab Jian-Jia kolm päeva selleks, et liikuda linnast 3 linna 0 ning külastab seitsmendal päeval 10 vaatamisväärsust linnas 1. Külastatud vaatamisväärsuste koguarv on 20 + 30 + 10 = 60, mis on maksimaalne vaatamisväärsuste arv, mida Jian-Jia saab 7 päevaga külastada, alustades linnast 2.

linn	vaatamisväärsuste arv
0	10
1	2
2	20
3	30
4	1

päev	tegevus
1	külastada linna 2 vaatamisväärsusi
2	liikuda linnast 2 linna 3
3	külastada linna 3 vaatamisväärsusi
4	liikuda linnast 3 linna 2
5	liikuda linnast 2 linna 1
6	liikuda linnast 1 linna 0
7	külastada linna 0 vaatamisväärsusi

Ülesanne

Realiseerida funktsioon `findMaxAttraction`, mis leiab maksimaalse vaatamisväärsuste arvu, mida Jian-Jia saab külastada.

- `findMaxAttraction(n, start, d, attraction)`
  - `n`: linnade arv.
  - `start`: alguslinna number.
  - `d`: päevade arv.
  - `attraction`: massiiv pikkusega n; `attraction[i]` on vaatamisväärsuste arv linnas i, kus $0 \leq i \leq n-1$.
  - Funktsioon peab tagastama maksimaalse vaatamisväärsuste arvu, mida Jian-Jia saab külastada.

Alamülesanded

Kõigis alamülesannetes $0 \leq d \leq 2n + \lfloor n/2 \rfloor$ ning vaatamisväärtuste arv on mittenegatiivne.

Täiendavad piirangud:

Alamülesanne	punkte	n	maksimaalne vaatamisväärsuste arv linnas	alguslinn
1	7	$2 \leq n \leq 20$	1 000 000 000	pole piiranguid
2	23	$2 \leq n \leq 100\,000$	100	linn 0
3	17	$2 \leq n \leq 3000$	1 000 000 000	pole piiranguid
4	53	$2 \leq n \leq 100\,000$	1 000 000 000	pole piiranguid

Realisatsiooni detailid

Esitada tuleb täpselt üks fail nimega `holiday.c`, `holiday.cpp` või `holiday.pas`. Selles failis peab olema eelpool kirjeldatud alamprogramm järgmise signatuuriga. C/C++ programmis tuleb kaasata ka päisfail `holiday.h`.

Pane tähele, et tulemus võib olla suur ning `findMaxAttraction` tagastab 64-bitise täisarvu.

C/C++ program

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

Pascal program

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

Näidishindaja

Näidishindaja loeb sisendit järgmises formaadis:

- Esimesel real: `n`, `start`, `d`.
- Teisel real: `attraction[0]`, ..., `attraction[n-1]`.

Näidishindaja väljastab funktsiooni `findMaxAttraction` tagastatud väärtuse.