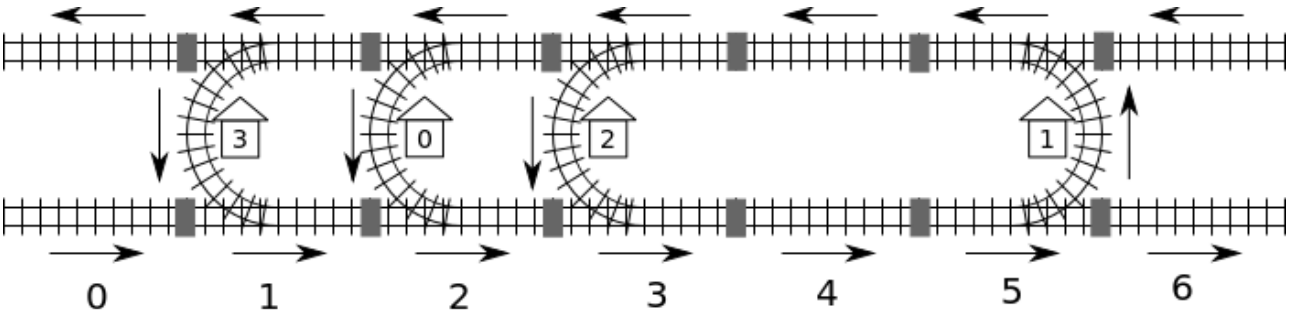
Železnica

Tajvan ima dolgo železniško progo, ki povezuje zahodni in vzhodni del otoka.
Ta železniška proga se sestoji iz m blokov. Zaporedni bloki so oštevilčeni s številkami $0, \ldots, m-1$, štetje pa se začne na zahodnem delu. Vsak blok ima en tir po katerem lahko potujemo samo v smeri vzhod-zahod in en tir po katerem lahko potujemo samo v smeri zahod-vzhod. Tir vzhod-zahod se nahaja na severnem delu bloka, tir zahod-vzhod pa na južnem delu bloka. Med tema dvema tiroma je lahko tudi železniška postaja.

Na progi obstajajo tri vrste blokov. Blok vrste *C* ima železniško postajo, na katero se vlaki pripeljejo preko severnega tira in jo zapustijo preko južnega tira. Blok vrste *D* ima tudi železniško postajo, vlaki pa nanjo pridejo z južnega tira in jo zapustijo preko severnega. Tretja vrsta blokov so *prazni* bloki, ki železniške postaje nimajo. Za primere teh treh vrst blokov si oglejmo spodnjo sliko. Bloki 1, 2 in 3 so vrste *C*, blok 5 je vrste *D* in bloki 0, 4 in 6 so *prazni*. Bloke povezujemo v progo tako, da sosednje tire povežemo s konektorji. Na spodnji sliki so konektorji prikazani z malimi osenčenimi pravokotniki.



Železniški sistem ima n postaj, oštevilčenih od 0 do $n-1$. Vedno lahko predpostavimo, da lahko s katere koli postaje pridemo do katere koli druge. Npr. na zgornji sliki lahko s postaje 0 pridemo do postaje 2 tako, da začnemo potovanje iz bloka 2, nadaljujemo pot preko blokov 3 in 4 po južnem tiru, se zapeljemo preko bloka 5 čez postajo 1, nato švignemo po severnem delu bloka 4 in končno prispemo na postajo 2, ki se nahaja v bloku 3.

Ker seveda lahko obstaja veliko različnih poti med dvema postajama, definirajmo razdaljo med dvema postajama kot *najmanjše* število konektorjev skozi katere gre pot. Npr. najkrajša pot med postajama 0 in 2 gre skozi bloke 2-3-4-5-4-3 in prevozi 5 konektorjev. Najkrajša razdalja med tema dvema postajama je torej 5.

Ta železniški sistem je nadzorovan z računalniškim programom, ki pa po hudem električnem mrku (žled?) ne ve niti kje se nahajajo postaje niti kakšni so tipi blokov na progi.
Edini podatek, ki je računalniku znan, je številka bloka v katerem se nahaja postaja 0. Ta postaja se vedno nahaja v bloku vrste *C*. Na srečo lahko računalnik vpraša kolikšna je razdalja med poljubnima dvema postajama. Npr., računalnik lahko vpraša: "Kolikšna je razdalja med postajama 0 in 2?". Odgovor na to vprašanje bo seveda 5.

Naloga

Imprementirajte funkcijo `findLocation`, ki za vsako postajo ugotovi številko bloka v katerem se nahaja in vrsto tega bloka.

- `findLocation(n, first, location, stype)`
  - `n`: število postaj.
  - `first`: številko bloka, v katerem se nahaja postaja 0.
  - `location`: tabela velikosti n; v `location[i]` zapišite številko bloka v katerem se nahaja postaja i.
  - `stype`: tabela velikosti n; v `stype[i]` zapiši vrsto bloka v katerem se nahaja postaja i: 1 za vrsto C in 2 za vrsto D.

Na voljo je funkcija `getDistance`, ki ti pomaga ugotoviti postavitve postaj in vrste blokov.

- `getDistance(i, j)` vrne razdaljo od postaje `i` do postaje `j`. Klic `getDistance(i, i)` vrne 0 (razdalja do iste postaje je seveda 0). Klic `getDistance(i, j)` vrne -1 če sta `i` ali `j` izven obsega $0 \leq i, j \leq n-1$.

Podnaloge

Pri vseh podnalogah je število blokov m največ 1,000,000. Pri nekaterih podnalogah je število klicev funkcije `getDistance` omejeno. Ta omejitev se med podnalogami razlikuje. Če program prekorači to omejitev, dobiš sporočilo o napačnem odgovoru ('wrong answer').

podnaloga	točke	n	št. klicev getDistance	opomba
1	8	$1 \leq n \leq 100$	neomejeno	Vse postaje (razen postaje 0) so v blokih vrste D.
2	22	$1 \leq n \leq 100$	neomejeno	Vse postaje desno od postaje 0 so v blokih vrste D, vse postaje levo od postaje 0 so v blokih vrste C.
3	26	$1 \leq n \leq 5,000$	$n(n-1)/2$	brez dodatnih omejitev
4	44	$1 \leq n \leq 5,000$	$3(n-1)$	brez dodatnih omejitev

Podrobnosti implementacije

Oddati moraš natanko eno datoteko poimenovano `rail.c`, `rail.cpp` ali `rail.pas`.
V tej datoteki implementiraj funkcijo `findLocation` natanko tako kot je opisana zgoraj. **V C/C++ morate obvezno vključiti tudi header datoteko `rail.h`.**

Podpis funkcije `getDistance` pa je sledeči:

V programskem jeziku C/C++

```
void findLocation(int n, int first, int location[], int stype[]);
```

V programskem jeziku Pascal

```
procedure findLocation(n, first : longint; var location,
stype : array of longint);
```

Vzorčni ocenjevalnik

Vzorčni ocenjevalnik bere vhod v sledečem formatu:

- vrstica 1: številka podnaloge
- vrstica 2: `n`
- vrstica $3+i$, ($0 \leq i \leq n-1$): `stype[i]` (1 za vrsto C in 2 za vrsto D), `location[i]`. Vzorčni ocenjevalnik bo izpisal `Correct`, če bodo `location[0]` ... `location[n-1]` in `stype[0]` ... `stype[n-1]`, ki jih je naračunal tvoj program, ustrezali vhodnim podatkom, sicer bo izpisal `Incorrect`.