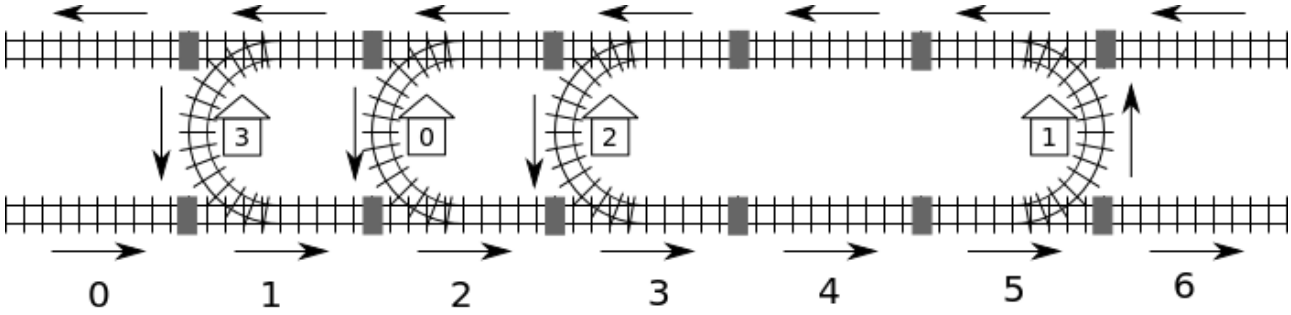
Rautatie

Taiwanissa on laaja rautatielinja, joka yhdistää saaren länsi- ja itärannikon. Linja muodostuu m pätkästä. Peräkkäiset pätkät on numeroitu $0, \ldots, m-1$ aloittaen lännestä päin. Jokaisessa pätkässä on yksi länteen kulkeva rataosuus pohjoisessa sekä yksi itään kulkeva rataosuus etelässä. Lisäksi rataosuuksien välissä on mahdollisesti juna-asema.

Pätkiä on kolmentyyppisiä. Tyypin *C* pätkässä on juna-asema, johon täytyy mennä pohjoisesta radasta ja josta täytyy poistua eteläiseen rataan. Tyypin *D* pätkässä on juna-asema, johon täytyy mennä eteläisestä radasta ja josta täytyy poistua pohjoiseen rataan. Tyypin *tyhjä* pätkässä ei ole juna-asemaa. Esimerkiksi seuraavassa kuvassa pätkät 0, 4 ja 6 ovat tyyppiä tyhjä, pätkät 1, 2 ja 3 ovat tyyppiä C ja pätkä 5 on tyyppiä D. Pätkät kytkeytyvät toisiinsa vaakasuuntaisesti. Vierekkäisten pätkien rataosuudet on yhdistetty liitoksilla, jotka näkyvät suorakulmioina seuraavassa kuvassa.



Ratajärjestemässä on n asemaa, jotka on numeroitu $0, \ldots, n-1$. Oletuksena on, että *voimme kulkea mistä tahansa asemasta mihin tahansa asemaan* rataa pitkin. Esimerkiksi voimme kulkea asemasta 0 asemaan 2 aloittamalla pätkästä 2, kulkemalla pätkiä 3 ja 4 eteläistä rataa, kulkemalla pätkässä 5 aseman 1 kautta, kulkemalla pätkää 4 pohjoista rataa ja lopettamalla reitti asemaan 2 pätkässä 3.

Koska reittimahdollisuuksia on useita, etäisyys asemasta toiseen on *pienin mahdollinen* määrä liitoksia reitillä. Esimerkiksi etäisyys asemasta 0 asemaan 2 kulkee pätkiä 2-3-4-5-4-3 ja ohittaa 5 liitosta, joten etäisyys on 5.

Tietokonejärjestelmä hallinnoi ratajärjestelmää. Valitettavasti sähkökatkon jälkeen tietokone ei enää tiedä, missä asemat sijaitsevat ja minkä tyyppisissä pätkissä ne ovat. Ainoa käytettävissä oleva tieto on, missä pätkässä asema 0 sijaitsee. Tämän pätkän tyyppi on aina C. Onneksi tietokone voi kysyä, mikä on etäisyys mistä tahansa asemasta mihin tahansa asemaan. Esimerkiksi tietokone voi kysyä 'mikä on etäisyys asemasta 0 asemaan 2?' ja saada vastauksen 5.

Tehtävä

Tehtäväsi on toteuttaa funktio 'findLocation', joka määrittää jokaisen aseman pätkän numeron ja tyypin.

- `findLocation(n, first, location, stype)`
  - `n`: asemien lukumäärä.
  - `first`: aseman 0 pätkän numero.
  - `location`: taulukko kokoa n; sinun tulee laittaa aseman i pätkän numero kohtaan `location[i]`.
  - `stype`: taulukko kokoa n; sinun tulee laittaa aseman i pätkän tyyppi kohtaan `stype[i]`: 1 tarkoittaa tyyppiä C ja 2 tarkoittaa tyyppiä D.

Voit kutsua funktiota `getDistance`, josta on apua asemien sijaintien ja tyyppien selvittämisessä.

- `getDistance(i, j)` palauttaa etäisyyden asemasta `i` asemaan `j`. `getDistance(i, i)` palauttaa 0. `getDistance(i, j)` palauttaa -1, jos `i` tai `j` ovat välin $0 \le i, j \le n-1$ ulkopuolella.

Osatehtävät

Kaikissa osatehtävissä pätkien määrä m on enintään $1\,000\,000$. Joissain osatehtävissä funktion `getDistance` kutsujen määrä on rajoitettu. Raja vaihtelee osatehtävän mukaan. Ohjelmasi saa viestin 'wrong answer', jos se ylittää tämän rajan.

osatehtävä	pisteet	n	getDistance-kutsuja	lisäehto
1	8	$1 \le n \le 100$	rajoittamaton	Kaikki asemat paitsi 0 ovat tyypin D pätkissä.
2	22	$1 \le n \le 100$	rajoittamaton	Kaikki asemat aseman 0 itäpuolella ovat tyypin D pätkissä ja kaikki asemat aseman 0 länsipuolella ovat tyypin C pätkissä.
3	26	$1 \le n \le 5\,000$	$n(n-1)/2$	ei lisäehtoa
4	44	$1 \le n \le 5\,000$	$3(n-1)$	ei lisäehtoa

Toteutus

Sinun täytyy lähettää tarkalleen yksi tiedosto, jonka nimi on `rail.c`, `rail.cpp` tai `rail.pas`. Tämä tiedosto toteuttaa funktion `findLocation` yllä olevan kuvauksen mukaisesti käyttäen seuraavia runkoja. Sinun täytyy myös lisätä mukaan otsikkotiedosto `rail.h` C/C++-toteutusta varten.

C/C++-ohjelma

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

Pascal-ohjelma

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

Funktion `getDistance` runko on seuraava.

C/C++-ohjelma

```
int getDistance(int i, int j);
```

Pascal-ohjelma

```
function getDistance(i, j: longint): longint;
```

Esimerkkitarkastin

Esimerkkitarkastin lukee syötteen seuraavassa muodossa:

- rivi 1: osatehtävän numero
- rivi 2: `n`
- rivi $3+i$, ($0 \leq i \leq n-1$): `stype[i]` (1 tarkoittaa tyyppiä C ja 2 tarkoittaa tyyppiä D), `location[i]`.

Esimerkkitarkastin tulostaa `Correct`, jos `location[0]` ... `location[n-1]` ja `stype[0]` ... `stype[n-1]` vastaavat syötettä funktion `findLocation` päättyessä tai `Incorrect`, jos ne eivät vastaa.