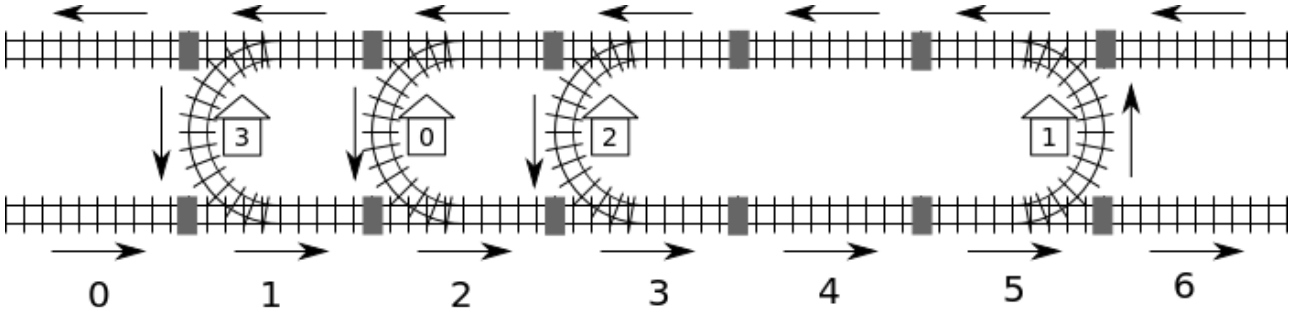
Raudtee

Taiwanil on pikk lääne- ja idakallast ühendav raudtee. Raudtee koosneb m lõigust, mis on nummerdatud $0, \ldots, m-1$ läänest ida suunas. Igal lõigul on põhja pool üks ühesuunaline rööpapaar, millel rongid võivad sõita ainult idast läände ja lõuna pool teine ühesuunaline rööpapaar, millel rongid võivad sõita ainult läänest itta; mõnel lõigul on nende kahe rööpapaari vahel lisaks ka raudteejaam.

Teelõike on kolme tüüpi. *C*-tüüpi lõigul on jaam, millesse rongid saavad siseneda põhjapoolselt rööpapaarilt ja millest väljuda lõunapoolsele rööpapaarile. *D*-tüüpi lõigul olevasse jaama saavad rongid siseneda lõunast ja sellest väljuda põhja. Kolmandat tüüpi nimetame *tühjaks*, sest seal jaama ei ole. Näiteks alloleval joonisel on lõigud 0, 4 ja 6 tühjad, lõigud 1, 2 ja 3 C-tüüpi ning lõik 5 D-tüüpi. Lõigud on omavahel ühendatud horisontaalselt ja kahe naaberlõigu rööpapaaride vahel on ühenduslülid, mis on joonisel näidatud hallide ristkülikutena.



Raudteel on kokku n jaama, mis on nummerdatud $0, \ldots, n-1$. Võib eeldada, et *igast jaamast on võimalik sõita igasse teise*. Näiteks saab jaamast 0 jaama 2 sõita järgmiselt: alustame lõigust 2, sõidame mööda lõunapoolseid rööpaid läbi lõikude 3 ja 4, seejärel lõigus 5 läbi jaama 1 põhjapoolsetele rööbastele ja neid mööda läbi lõigu 4, kuni jõuame lõigul 3 jaama 2.

Kuna võimalikke viise ühest jaamast teise sõita on mitu, loeme kahe jaama vaheliseks kauguseks *minimaalse* ühenduslülide arvu, mis tuleb läbida teel esimesest jaamast teise. Näiteks lühim tee jaamast 0 jaama 2 on läbi lõikude 2-3-4-5-4-3 ja läbib 5 ühenduslüli; seega on jaamade 0 ja 2 vaheline kaugus 5.

Rongide liiklust raudteel juhib arvuti. Kahjuks ei mäleta arvuti pärast voolukatkestust enam jaamade asukohti ja nende tüüpe. Ainus, mida arvuti mäletab, on jaama 0 asukoht, ja et see on C-tüüpi lõigul. Õnneks on arvutil alles funktsioon, mis tagastab mistahes kahe jaama vahelise kauguse. Näiteks võib arvuti esitada päringu 'milline on jaamade 0 ja 2 vaheline kaugus?' ja saada vastuseks 5.

Ülesanne

Realiseerida alamprogramm `findLocation`, mis leiab iga jaama asukohalõigu numbri ja tüübi.

- `findLocation(n, first, location, stype)`
  - `n`: jaamade arv;

- `first`: jaama 0 asukoht (lõigu number);
- `location`: n elemendiga massiiv; `findLocation` peab elementi `location[i]` salvestama jaama i asukohalõigu numbri;
- `stype`: n elemendiga massiiv; `findLocation` peab elementi `stype[i]` salvestama jaama i asukohalõigu tüübi (1, kui lõik on C-tüüpi; 2, kui lõik on D-tüüpi).

Alamprogramm `findLocation` saab jaamadevaheliste kauguste leidmiseks kasutada funktsiooni `getDistance`.

- `getDistance(i, j)` tagastab kauguse jaamast `i` jaama `j`. `getDistance(i, i)` tagastab 0. `getDistance(i, j)` tagastab -1, kui `i` või `j` on väljaspool lubatud piire $0 \le i, j \le n - 1$.

Alamülesanded

Teelõikude arv m ei ületa üheski alamülesandes $1\,000\,000$. Mõnes alamülesandes on funktsiooni `getDistance` väljakutsete arv piiratud (piir on erinevates alamülesannetes erinev). Kui lahendus ületab lubatud väljakutsete arvu, on serveri diagnoos 'Wrong answer'.

alamülesanne	punkte	n	`getDistance` kutseid	märkused
1	8	$1 \le n \le 100$	piiramatult	kõik jaamad peale jaama 0 on D-tüüpi lõikudel
2	22	$1 \le n \le 100$	piiramatult	kõik jaamast 0 ida pool olevad jaamad on D-tüüpi ja kõik lääne pool olevad jaamad C-tüüpi lõikudel
3	26	$1 \le n \le 5\,000$	$n(n-1)/2$	lisatingimusi ei ole
4	44	$1 \le n \le 5\,000$	$3(n-1)$	lisatingimusi ei ole

Realisatsiooni detailid

Lahendusena tuleb esitada täpselt üks fail nimega `rail.c`, `rail.cpp` või `rail.pas`. Selles failis peab olema eelpool kirjeldatud alamprogramm `findLocation` täpselt järgneva signatuuriga. C ja C++ lahendused peavad lisaks kaasama päisfaili `rail.h`.

C ja C++

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

Pascal

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

Funktsiooni `getDistance` signatuur on järgnev.

C ja C++

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

Pascal

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

Näidishindaja

Näidishindaja eeldab järgmist sisendi vormingut:

- Esimesel real: alamülesande number.
- Teisel real: arv `n`.
- Real $3+i$, kus ($0 \leq i \leq n-1$): arvud `stype[i]` (1 C-tüüpi ja 2 D-tüüpi lõigu korral) ja `location[i]`.

Näidishindaja väljastab `Correct`, kui alamprogrammi `findLocation` leitud `location[0]` ... `location[n-1]` ja `stype[0]` ... `stype[n-1]` väärtused vastavad sisendile, ja `Incorrect`, kui ei vasta.