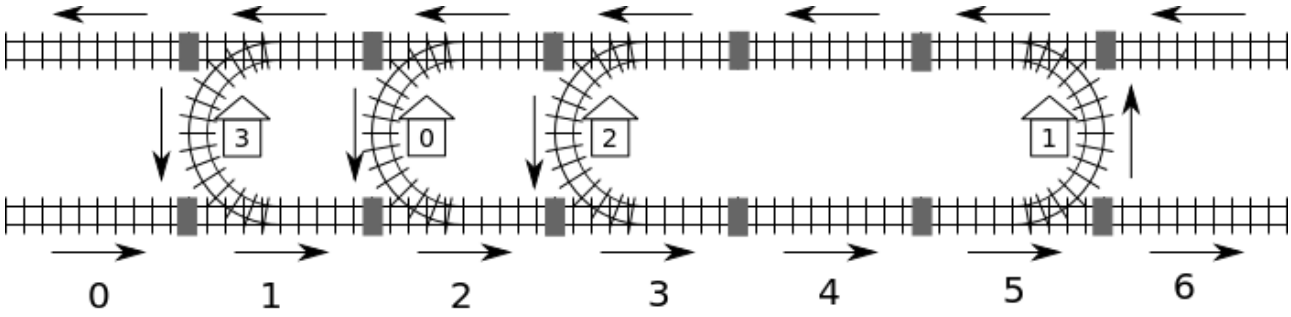
# Sliedes

Taivānā ir liela dzelzceļa līnija, kas savieno salas rietumu un austrumu piekrastes. Līnija sastāv no $m$ *blokiem*, kas sanumurēti ar skaitļiem no $0$ līdz $m-1$ pēc kārtas, sākot no rietumu piekrastes. Katrs bloks satur divus vienvirziena sliežu posmus: ziemeļu posmu, kurā kustība notiek no austrumiem uz rietumiem, un dienvidu posmu, kurā kustība notiek no rietumiem uz austrumiem. Papildus, bloks var saturēt *staciju* starp tā sliežu posmiem.

Ir iespējami trīs veidu bloki:

- *C* tipa blokam ir stacija, kurā iebrauc no ziemeļu posma un no kuras izbrauc uz dienvidu posmu;
- *D* tipa blokam ir stacija, kurā iebrauc no dienvidu posma un no kuras izbrauc uz ziemeļu posmu;
- *tukša* tipa blokam nav stacijas.

Piemēram, zemāk redzamajā attēlā bloki nr.0, nr.4 un nr.6 ir tukšā tipa, bloki nr.1, nr.2 un nr.3 ir C tipa, bet bloks nr.5 ir D tipa. Blakus bloki (attiecīgie sliežu posmi) ir savienoti ar *savienojumiem*, kas attēloti kā tumšāki taisnstūri:




Pavisam ir $n$ stacijas, kas sanumurētas no $0$ līdz $n-1$. Ir zināms, ka *no jebkuras stacijas var nokļūt uz jebkuru citu staciju*, braucot pa sliežu posmiem. Piemēram, no stacijas 0 ir iespējams nokļūt uz staciju 2, ja no bloka nr.2 pa dienvidu posmiem brauc caur blokiem nr.3 un nr.4, tad blokā nr.5 izbrauc caur staciju 1, tālāk pa ziemeļu posmu izbrauc caur bloku nr.4, līdz beidzot blokā nr.3 sasniedz staciju 2.

Tā kā ir iespējami vairāki dažādi ceļi, attālums no vienas stacijas līdz otrai tiek definēts kā *mazākais* savienojumu skaits, ko šķērso ceļš. Piemēram, īsākais ceļš no stacijas 0 uz staciju 2 ir caur blokiem 2-3-4-5-4-3 un šķērso 5 savienojumus, tātad attālums ir 5.

Dzelzceļu vada datorsistēma. Diemžēl pēc elektroapgādes traucējumiem tika pazaudēta informācija par to, kur un kāda veida blokos atrodas stacijas. Vienīgi zināms, ka stacija 0 vienmēr atrodas C tipa blokā. Par laimi ir iespējams veikt vaicājumus par attālumu no jebkuras stacijas līdz jebkurai citai stacijai. Piemēram, uz vaicājumu 'kāds ir attālums no stacijas 0 līdz stacijai 2?' tiks saņemta atbilde 5.

# Uzdevums

Jums jāuzraksta funkcija `findLocation`, kas katrai stacijai noskaidro, kurā blokā tā atrodas un kāds

ir šī bloka tips.

- `findLocation(n, first, location, stype)`
  - `n`: staciju skaits.
  - `first`: bloka numurs, kurā atrodas stacija 0.
  - `location`: $n$ elementu masīvs; elementā `location[i]` jums jāieraksta bloka, kurā atrodas stacija ar numuru $i$, numurs.
  - `stype`: $n$ elementu masīvs; elementā `stype[i]` jums jāieraksta bloka, kurā atrodas stacija ar numuru $i$, tips. Masīva elementā jāieraksta skaitlis 1, ja bloks ir C tipa, vai 2, ja bloks ir D tipa.

Jūs varat izsaukt funkciju `getDistance`, lai palīdzētu noskaidrot bloku, kuros atrodas stacijas, numurus un tipus.

- `getDistance(i, j)` atgriež attālumu no stacijas `i` līdz stacijai `j`. Izsaukums `getDistance(i, i)` atgriezīs 0, bet `getDistance(i, j)` atgriezīs -1, ja `i` vai `j` ir ārpus pieļaujamo vērtību apgabala $0 \leq i, j \leq n-1$.

# Apakšuzdevumi

Visos apakšuzdevumos bloku skaits $m$ nepārsniedz 1,000,000. Dažos apakšuzdevumos ir ierobežots atļautais `getDistance` funkcijas izsaukumu skaits, kas ir atšķirīgs dažādiem apakšuzdevumiem. Pārsniedzot šo ierobežojumu, apakšuzdevuma rezultāts būs 'nepareiza atbilde' ('wrong answer').

| apakšuzdevums | punkti | $n$ | getDistance izsaukumu skaits | piezīmes |
|---|---|---|---|---|
| 1 | 8 | $1 \leq n \leq 100$ | neierobežots | Visas stacijas, izņemot staciju 0, atrodas D tipa blokos. |
| 2 | 22 | $1 \leq n \leq 100$ | neierobežots | Visas stacijas uz austrumiem no stacijas 0 atrodas D tipa blokos, un visas stacijas uz rietumiem no stacijas 0 atrodas C tipa blokos. |
| 3 | 26 | $1 \leq n \leq 5,000$ | $n(n-1)/2$ | bez papildus ierobežojumiem |
| 4 | 44 | $1 \leq n \leq 5,000$ | $3(n-1)$ | bez papildus ierobežojumiem |

# Implementācijas detaļas

Jums jaiesūta tieši viens fails ar nosaukumu `rail.c`, `rail.cpp` vai `rail.pas`. Šim failam jārealizē `findLocation` funkcija, kas aprakstīta iepriekš, ar norādīto signatūru. C/C++ programmas gadījumā Jums programmas tekstā jāiekļauj `rail.h`.

## C/C++ programma

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

## Pascal programma

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

Funkcijas `getDistance` signatūra ir šāda:

## C/C++ programma

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

## Pascal programma

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

## Paraugtestētājs

Paraugtestētājs lasa ievadu šādā formātā:

- rinda 1: apakšuzdevuma numurs
- rinda 2: `n`
- rinda $3 + i$, ($0 \leq i \leq n - 1$): `stype[i]` (C tipa bloks tiek apzīmēts ar skaitli 1, bet D tipa bloks - ar skaitli 2), `location[i]`.

Paraugtestētāja atbilde būs `Correct`, ja jūsu programmas aprēķinātās `location[0]` ... `location[n-1]` un `stype[0]` ... `stype[n-1]` vērtības atbilst ievaddatiem, kad `findLocation` beidz darbu, vai `Incorrect`, ja vērtības neatbilst.