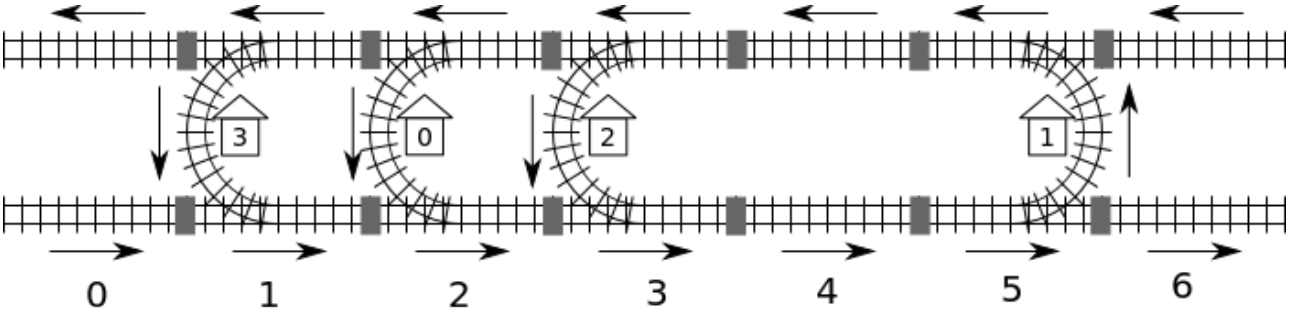
Spoorlijn

Taiwan heeft een grote spoorlijn die de west- en oostkust van het eiland verbindt. De lijn bestaat uit m secties. De opeenvolgende secties zijn genummerd $0, \ldots, m-1$, beginnend vanaf het westelijke eindpunt. Elke sectie heeft aan de noordkant een enkel spoor richting het westen, en aan de zuidkant een enkel spoor richting het oosten, met optioneel een station daartussen.

Er zijn drie types secties. Een type *C* sectie heeft een station dat je moet binnenrijden vanaf het noordelijke spoor en moet buitenrijden naar het zuidelijke spoor. Een type *D* sectie heeft een station dat je moet binnenrijden vanaf het zuidelijke spoor en buitenrijden naar het noordelijke spoor. Een type *leeg* sectie heeft geen station. Bijvoorbeeld, in de figuur hieronder zijn secties 0, 4 en 6 van type *leeg*, secties 1, 2 en 3 van type *C*, en sectie 5 is type *D*. De sporen van naast elkaar liggende secties zijn verbonden met *connectors*, die in de figuur door grijze rechthoekjes worden voorgesteld.



Het spoorsysteem heeft n stations genummerd van 0 tot $n-1$. We nemen aan dat *we van elk station naar eender welk ander station kunnen gaan* door het spoor te volgen. We kunnen bijvoorbeeld van station 0 naar station 2 gaan door te starten in sectie 2, dan te passeren door sectie 3 en 4 via het zuidelijke spoor, vervolgens sectie 5 te passeren door station 1, dan terug door sectie 4 te gaan via het noordelijke spoor, om tenslotte station 2 te bereiken in sectie 3.

Gezien er verschillende mogelijke routes zijn, definiëren we de afstand tussen twee stations als het *minimum* aantal connectors waar de route doorheen loopt. Bijvoorbeeld: de kortste route van station 0 naar station 2 is doorheen de secties 2-3-4-5-4-3 en passeert zo door 5 connectors, dus de afstand is 5.

Het spoorsysteem wordt beheerd door een computer. Helaas weet die na een stroomuitval niet meer waar de stations zijn en in welk type sectie ze gelegen zijn. De enige hint die de computer nog heeft is het sectienummer van station 0, dat altijd in een type C sectie ligt. Gelukkig kan de computer wel opvragen wat de afstand is van elk station tot elk ander station. Bijvoorbeeld, de computer kan opvragen 'wat is de afstand van station 0 tot station 2?' en het zal als antwoord 5 krijgen.

Taak

Je moet een functie `findLocation` implementeren die voor elk station bepaalt wat het sectienummer en het sectietype is.

- `findLocation(n, first, location, stype)`
  - `n`: het aantal stations.
  - `first`: het sectienummer van station 0.
  - `location`: array van lengte n; je moet het sectienummer van station i in `location[i]` schrijven.
  - `stype`: array van lengte n; je moet het sectietype van station i in `stype[i]` schrijven: 1 voor type C en 2 voor type D.

Je kan een functie `getDistance` aanroepen om je te helpen de locaties en types van stations te vinden:

- `getDistance(i, j)` geeft de afstand van station `i` tot station `j` terug. `getDistance(i, i)` geeft 0 terug. `getDistance(i, j)` geeft -1 terug als `i` of `j` buiten het bereik $0 \leq i, j \leq n-1$ liggen.

Subtaken

In alle subtaken is het aantal secties m niet groter dan 1,000,000. In sommige subtaken kan je maar een beperkt aantal keer `getDistance` oproepen. De limiet varieert in elke subtaak. Je programma zal 'wrong answer' teruggeven als deze limiet wordt overschreden.

subtaak	punten	n	getDistance oproepen	opmerking
1	8	$1 \leq n \leq 100$	ongelimiteerd	Alle stations behalve 0 liggen in type D secties.
2	22	$1 \leq n \leq 100$	ongelimiteerd	Alle stations ten oosten van station 0 liggen in type D secties, en alle stations ten westen van station 0 liggen in type C secties.
3	26	$1 \leq n \leq 5,000$	$n(n-1)/2$	geen andere beperkingen
4	44	$1 \leq n \leq 5,000$	$3(n-1)$	geen andere beperkingen

Implementatiedetails

Je moet exact één bestand indienen, genaamd `rail.c`, `rail.cpp` of `rail.pas`. Dit bestand implementeert `findLocation` zoals hierboven beschreven, volgens de declaratie hieronder. Bij C/C++ implementatie moet dat ook een header-bestand `rail.h` "includeren".

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);
```

De declaratie van `getDistance` is als volgt.

C/C++ programma

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

Pascal programma

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

Voorbeeldgrader

De voorbeeldgrader leest input in het volgende formaat:

- lijn 1: het nummer van de subtaak
- lijn 2: `n`
- lijnen $3+i$, ($0 \leq i \leq n-1$): `stype[i]` (1 voor type C en 2 voor type D), `location[i]`.

De voorbeeldgrader zal `Correct` weergeven als de`location[0]` ... `location[n-1]` en `stype[0]` ... `stype[n-1]` berekend door jouw programma overeenkomen met de input wanneer `findLocation` afgelopen is, of `Incorrect` als ze niet overeenkomen.