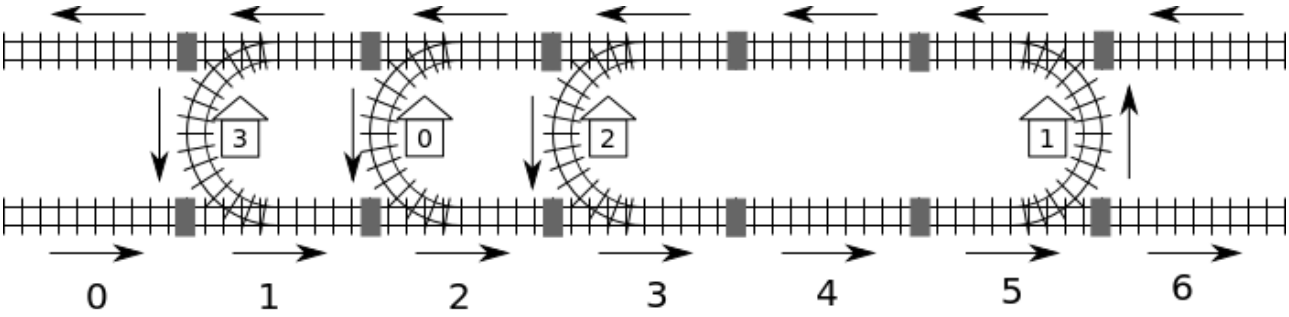
Rail

Taiwan heeft een spoorbaan die de westkust en oostkust van het eiland met elkaar verbindt. De spoorbaan bestaat uit m blokken. De blokken zijn van west naar oost genummerd van $0, \ldots, m-1$. Elk blok heeft aan de noordkant een stuk spoor in westelijke richting. Elk blok heeft een stuk spoor in oostelijke richting aan de zuidkant. Eventueel bevindt zich in een blok ook een station.

Er zijn blokken van drie types. Een type *C* heeft een station dat je vanuit het noorden inrijdt en aan de zuidkant weer verlaat. Een type *D* heeft een station dat je vanuit het zuiden inrijdt en aan de noordkant weer verlaat. Een type *empty* heeft helemaal geen station.

In het volgende voorbeeld zijn de blokken 0, 4 en 6 van type empty; blok 1, 2 en 3 zijn van type C en blok 5 is van type D. Blokken zijn horizontaal met elkaar verbonden. De rails van twee aangrenzende blokken zijn met twee verbindingsstukjes met elkaar verbonden. Deze zie je in het plaatje als grijze rechthoeken.



De spoorbaan heeft n stations die genummerd zijn van 0 tot en met $n-1$. Ga ervan uit dat je van elk station naar elk andere station kunt komen. Je kunt bijvoorbeeld van station 0 als volgt naar station 2 gaan: je begint in blok 2; dan ga je over de zuidelijke rails door blokken 3 en 4, dan door blok 5 en daarin door station 1, dan over de noordelijke rails door blok 4, en zo kom je tenslotte aan op station 2, welke ligt in blok 3.

Omdat er meerdere mogelijke routes zijn is de afstand tussen twee stations gedefinieerd als het *minimale* aantal verbindingsstukjes waar de route overheen gaat. De kortste route van station 0 naar station 2 is door blokken 2-3-4-5-4-3 en gaat door 5 verbindingsstukje. De afstand is dus 5.

Een computer beheert het spoorsysteem. Na een stroomstoring is de computer vergeten waar de stations liggen en in wat voor soort blok ze liggen. De computer weet nog wel in welk blok station 0 ligt, en dat dit een blok van type C is.

De computer kan de afstand tussen elk tweetal stations opvragen. De computer kan bijvoorbeeld vragen 'wat is de afstand van station 0 naar station 2'. Het antwoord is dan 5.

Task

Implementeer een functie `findLocation` die voor elk station bepaalt in welk blok nummer en wat

voor blok type het station ligt.

- `findLocation(n, first, location, stype)`
  - `n`: het aantal stations.
  - `first`: het blok nummer van station 0.
  - `location`: array van grootte n; sla het blok nummer van station i op in `location[i]`.
  - `stype`: array van grootee n; sla het blok type van station i op in `stype[i]`: gebruik 1 voor type C; gebruik 2 voor type D.

Je kunt de functie `getDistance` aanroepen om je te helpen de locaties en types van de stations te bepalen.

- `getDistance(i, j)` levert als resultaat de afstand van station `i` naar station `j`. `getDistance(i, i)` levert 0. `getDistance(i, j)` levert -1 als `i` of `j` valt buiten de voorwaarden $0 \leq i, j \leq n-1$.

Subtasks

Voor alle subtasks geldt dat het aantal station m niet groter is dan 1 000 000. In sommige subtasks mag je `getDistance` maar een beperkt aantal keren aanroepen. Deze limiet varieert per subtask. Je programma krijgt een 'wrong answer' als je boven de limiet uitkomt.

subtask	punten	n	getDistance aanroepen	opmerkingen
1	8	$1 \leq n \leq 100$	ongelimiteerd	Alle stations, behalve type 0 liggen in een type D blok.
2	22	$1 \leq n \leq 100$	ongelimiteerd	Alle stations ten oosten van station 0 liggen in een type-D blok. Alle stations ten westen van station 0 liggen in een type-C blok.
3	26	$1 \leq n \leq 5000$	$n(n-1)/2$	geen aanvullende beperkingen
4	44	$1 \leq n \leq 5000$	$3(n-1)$	geen aanvullende beperkingen

Implementatie details

Je moet precies één bestand inzenden met naam `rail.c`, `rail.cpp` of `rail.pas`. In deze file staat een implementatie van `findLocation` zoals hierboven is beschreven. Bij de C/++ implementatie moet je ook #include `rail.h` doen.

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);
```

`getDistance` moet je als volgt definieren:

C/C++ programma

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

Pascal programma

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

Voorbeeld grader

De voorbeeld grader verwacht invoer in het volgende formaat:

- regel 1: the subtask number
- regel 2: `n`
- regel $3 + i$, ($0 \leq i \leq n - 1$): `stype[i]` (1 betekent type C en 2 betekent type D), `location[i]`.

Wanneer `findLocation` klaar is drukt de voorbeeld grader `Correct` af als `location[0]` ... `location[n-1]` en `stype[0]` ... `stype[n-1]`, zoals door jouw berekend zijn, aansluiten bij de invoer. Anders drukt hij `Incorrect` af.