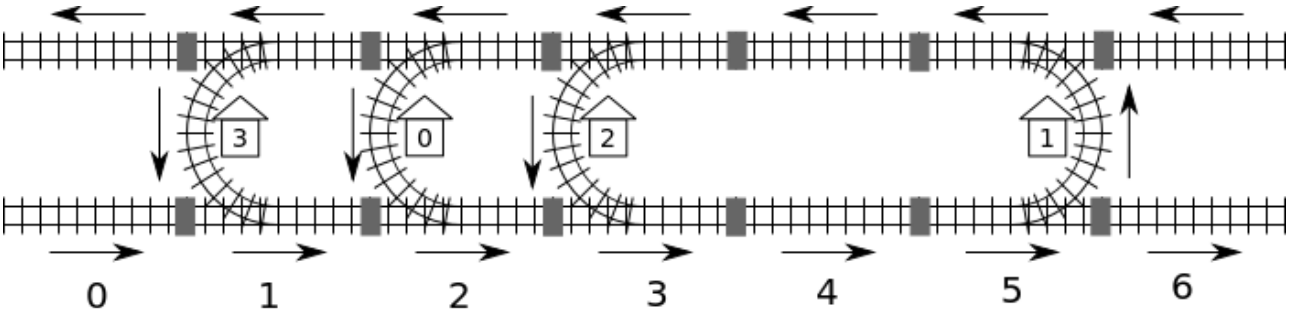
Rail

Taiwan mempunyai sebuah jalur besar kereta api yang menghubungkan tepian Barat dan tepian Timur pulaunya. Jalur kereta api tersebut terdiri dari m blok. Blok-blok yang berturutan tersebut dinomori $0, \ldots, m-1$, dimulai dari ujung paling Barat. Setiap blok mempunyai jalur satu arah dengan ujung akhir di sebelah Barat yang membentang di Utara, sebuah jalur satu arah dengan ujung akhir di sebelah Timur yang membentang di Selatan, dan mungkin ada stasiun kereta api di antaranya.

Ada tiga tipe blok. Blok tipe *C* mempunyai sebuah stasiun kereta api yang harus dimasuki dari jalur bagian Utara dan keluar di jalur bagian Selatan, blok tipe *D* mempunyai sebuah stasiun kereta api yang harus dimasuki dari jalur Selatan dan keluar dari jalur Utara, dan tipe blok *empty* tidak mempunyai stasiun kereta api. Misalnya, pada gambar berikut, blok 0, blok 4, dan blok 6 adalah blok bertipe *empty*; blok 1, blok 2, dan blok 3 adalah blok bertipe *C*; dan blok 5 adalah blok bertipe *D*. Jalur dari dua blok yang bertetangga secara langsung dihubungkan oleh *connector*, ditunjukkan dengan sejumlah persegi panjang diarsir (*shaded rectangles*) dalam gambar berikut.



Sistem rel kereta api mempunyai n stasiun dinomori dari 0 sampai dengan $n-1$. Diasumsikan, bahwa *kita dapat bepergian dari stasiun manapun ke stasiun lain manapun* dengan mengikuti jalur. Misalnya, kita dapat pergi dari stasiun 0 ke stasiun 2 mulai dari blok 2, kemudian lewat blok 3 dan blok 4 di jalur Selatan, dan kemudian melewati stasiun 1 yang berada di blok 5, kemudian melalui blok 4 jalur Utara, dan berakhir di stasiun 2 pada blok 3.

Karena ada banyak kemungkinan rute, jarak dari sebuah stasiun ke sebuah stasiun lainnya didefinisikan sebagai banyaknya *connector* yang *minimum* yang dilewati rute tersebut. Misalnya, rute terpendek dari stasiun 0 ke stasiun 2 adalah lewat blok 2-3-4-5-4-3 dan melalui 5 *connector*, sehingga jaraknya adalah 5.

Sebuah sistem komputer mengelola sistem rel kereta api tersebut. Malangnya, setelah terjadi suatu pemadaman listrik, komputer tidak lagi mengetahui lokasi stasiun dan tipe dari blok dimana stasiun tersebut berada. Satu-satunya petunjuk yang dimiliki oleh komputer adalah nomor blok dari stasiun 0, yang selalu merupakan blok tipe *C*. Untungnya, komputer dapat menanyakan jarak dari sebarang stasiun manapun ke sebarang stasiun lainnya. Misalnya, komputer dapat menanyakan *'what is the distance from station 0 to station 2?'* dan komputer akan mendapat jawaban 5.

Task

Anda perlu mengimplementasi sebuah function `findLocation` yang menentukan nomor-nomor blok dan tipe-tipe blok dari setiap stasiun.

- `findLocation(n, first, location, stype)`
  - `n`: the number of stations.
  - `first`: the block number of station 0.
  - `location`: array of size n; you should place the block number of station i into `location[i]`.
  - `stype`: array of size n; you should place the block type of station i into `stype[i]`: 1 for type C and 2 for type D.

Anda dapat memanggil (*call*) sebuah function `getDistance` yang membantu Anda untuk menemukan lokasi-lokasi dan tipe-tipe blok dari stasiun-stasiun.

- `getDistance(i, j)` returns the distance from station `i` to station `j`.
  `getDistance(i, i)` will return 0.
  `getDistance(i, j)` will return -1 if `i` or `j` is outside the range $0 \le i, j \le n-1$.

Subtasks

Pada semua subtask banyaknya blok-blok m tidak lebih dari 1,000,000. Pada beberapa subtask banyaknya pemanggilan (*call*) function `getDistance` dibatasi.
Batasan tersebut bervariasi sesuai dengan subtask. Program anda akan menerima 'wrong answer' jika melampaui batasan tersebut.

subtask	points	n	getDistance calls	note
1	8	$1 \le n \le 100$	unlimited	All stations except 0 are in type D blocks.
2	22	$1 \le n \le 100$	unlimited	All stations to the right of station 0 are in type D blocks, and all stations to the left of station 0 are in type C blocks.
3	26	$1 \le n \le 5,000$	$n(n-1)/2$	no additional limits
4	44	$1 \le n \le 5,000$	$3(n-1)$	no additional limits

Implementation details

Anda harus mensubmit satu file, dengan nama `rail.c`, `rail.cpp` atau `rail.pas`. File ini mengimplementasi function `findLocation` sebagaimana dijelaskan diatas, menggunakan *signature* sebagai berikut. Untuk implementasi C/C++, anda juga perlu meng-*include* sebuah header file `rail.h`.

C/C++ program

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

Pascal program

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

The signatures of `getDistance` are as follows.

C/C++ program

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

Pascal program

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

Sample grader

The sample grader reads the input in the following format:

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

The sample grader will print `Correct` if `location[0]` ... `location[n-1]` and `stype[0]` ... `stype[n-1]` computed by your program match the input when `findLocation` returns, or `Incorrect` if they do not match.