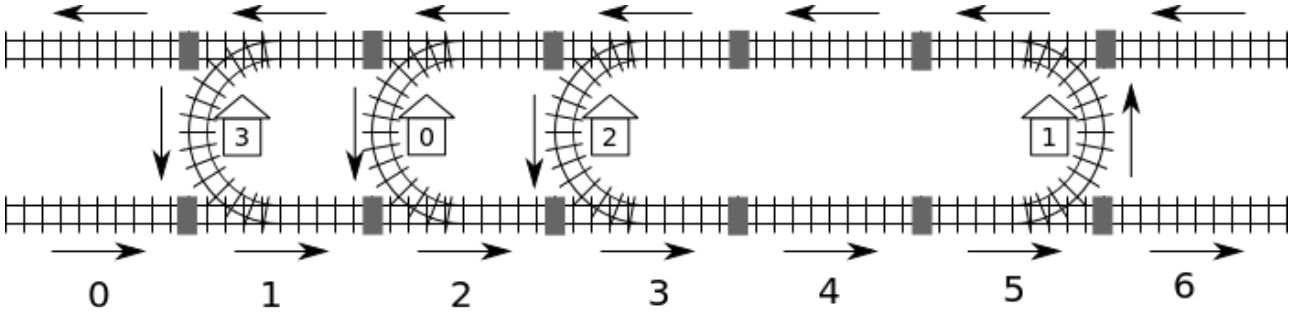
Колія

У Тайвані є велика залізнична лінія, що з'єднує західне за східне узбережжя острова. Лінія складається з m блоків. Послідовні блоки пронумеровані $0, \ldots, m-1$, починаючи з західного кінця. Кожен блок має на півночі колію з дозволеним рухом на захід, на півдні колію з дозволеним рухом на схід, та, можливо, станцію між ними.

Є три типи блоків. Блок типу *C* містить станцію, на яку ви маєте заїжджати з північної колії, та виїжджати по південній колії, блок типу *D* містить станцію, на яку ви маєте заїжджати з південної колії, та виїжджати по північній колії, блок типу *пустий* не містить станції. Наприклад, на наступному рисунку блоки 0, 4 та 6 мають тип пустий, блоки 1, 2 та 3 мають тип C, та блок 5 має тип D. Колії сусідніх блоків поєднано *з'єднувачами*, позначеними на рисунку як затемнені прямокутники.



Колійна система має n станцій, пронумерованих від 0 до $n-1$. Ми припускаємо, що *ми можемо дістатись довільної станції з довільної іншої станції* слідуючи коліям. Наприклад, ми можемо дістатися зі станції 0 до станції 2 почавши з блоку 2, потім слідуючи через блоки 3 та 4 по південній колії, потім через блок 5, проїхавши через станцію 1, потім проїхавши через блок 4 по північній колії, та, нарешті, прибувши на станцію 2 у блоці 3.

Оскільки може бути кілька маршрутів, відстань від однієї станції до іншої визначається як *мінімальна* кількість з'єднувачів, через які проходить маршрут. Наприклад, найкоротший маршрут від станції 0 до станції 2 проходить через блоки 2-3-4-5-4-3 та проходить через 5 з'єднувачів, тобто відстань дорівнює 5.

Колійною системою керує комп'ютер. Нажаль, після збою живлення комп'ютер більше не знає, де знаходяться станції і в блоках якого типу вони розташовані. Єдине, що відомо комп'ютеру, це номер блоку для станції 0, який завжди має тип C. Нащастя, комп'ютер може запитувати відстань від довільної станції до довільної іншої станції. Наприклад, компьютер може запитати 'чому дорівнює відстань від станції 0 до станції 2?', та отримає відповідь 5.

Задача

Ви маєте реалізувати функцію `findLocation`, що визначить для кожної станції номер блока і тип блока.

- `findLocation(n, first, location, stype)`
  - `n`: кількість станцій.
  - `first`: номер блока станції 0.
  - `location`: масив довжини n; ви маєте записати номер блока станції i до `location[i]`.
  - `stype`: масив довжини n; ви маєте записати тип блока станції i до `stype[i]`: 1 для типу C та 2 для типу D.

Ви можете викликати функцію `getDistance`, що потрібно дня знаходження розташування та типу станцій.

- `getDistance(i, j)` повертає відстань від станції `i` до станції `j`. `getDistance(i, i)` поверне 0. `getDistance(i, j)` поверне -1, якщо `i` або `j` знаходиться за межами діапазону $0 \leq i, j \leq n-1$.

Підзадачі

У всіх підзадачах кількість блоків m не перевищує 1,000,000. У деяких підзадачах обмежено кількість викликів `getDistance`. Це обмеження відрізняється для підзадач. Ваша програма отримає 'wrong answer', якщо вона перевищіть це обмеження.

підзадача	балів	n	викликів getDistance	примітка
1	8	$1 \leq n \leq 100$	не обмежено	Всі станції за винятком 0 розташовано у блоках типу D.
2	22	$1 \leq n \leq 100$	не обмежено	Всі станції на схід від станції 0 розташовано у блоках типу D, та всі станції на захід від станції 0 розташовано у блоках типу C.
3	26	$1 \leq n \leq 5,000$	$n(n-1)/2$	немає інших обмежень
4	44	$1 \leq n \leq 5,000$	$3(n-1)$	немає інших обмежень

Деталі реалізації

Ви маєте відіслати тільки один файл, що має ім'я `rail.c`, `rail.cpp` або `rail.pas`. Цей файл реалізує `findLocation` як описано вище, використовуючи такі сигнатури. Також підключіть файл заголовків `rail.h` у програмі на C/C++.

Програма на C/C++

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

Програма на Pascal

```
procedure findLocation(n, first : longint; var location,
```

```
  stype : array of longint);
```

Сигнатура `getDistance` така:

Програма на C/C++

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

Програма на Pascal

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

Приклад модуля перевірки

Наданий вам модуль перевірки читає вхідні дані у наступному форматі:

- рядок 1: номер підзадачі
- рядок 2: `n`
- рядок $3+i$, ($0 \le i \le n-1$): `stype[i]` (1 для типу C та 2 для типу D), `location[i]`.

Модуль перевірки надрукує `Correct` якщо `location[0]` ... `location[n-1]` та `stype[0]` ... `stype[n-1]`, обчислені вашою програмою, співпадають після повернення з `findLocation` з вхідними даними, або `Incorrect`, якщо вони не співпадають.