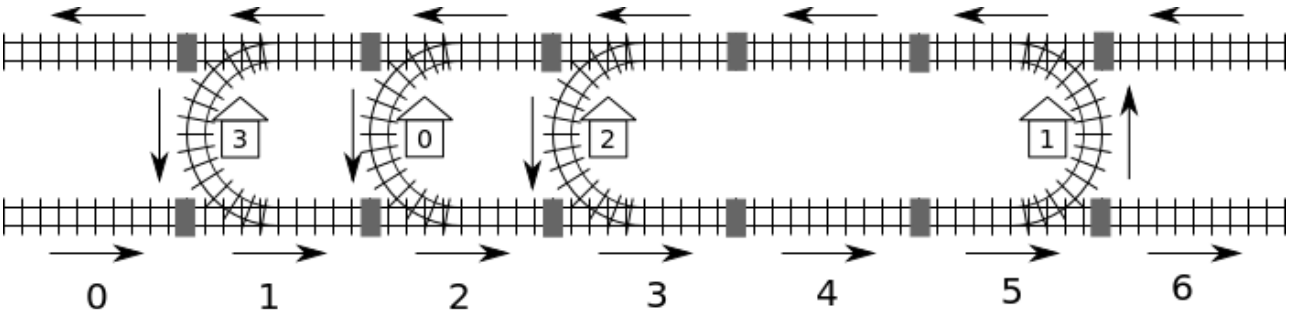
Rail

Тайван има една дълга железопътна мрежа, свързваща западния с източния бряг, която се състои от m блока, номерирани от 0 до $m-1$, започвайки от запад на изток. Всеки блок се състои от еднопосочна линия от изток на запад, еднопосочна линия от запад на изток, разположена южно от линията водеща от изток на запад, и може да има (или да няма) гара разположена, между двете еднопосочни линии и свързана с тях.

Възможни са три типа блокове. В блоковете от тип *C* има гара, в която се влиза от северната линия и се излиза към южната, в блоковете от тип *D* има гара, в която се влиза от южната линия и се излиза към северната, а блоковете от третия тип са *празни*, т.е. в тях няма гара. Например, блокът с номер 0 на фигурата е празен, блокът с номер 1 е от тип *C*, а блокът 5 е от тип *D*. Отделните блокове са свързани хоризонтално като северните линии на всеки от два съседни блока, както и южните линии на всеки два съседни блока са свързани една с друга с по един *конектор*. Конекторите са показани на фигурата със запълнени правоъгълници.



Броят на станциите на мрежата е n, номерирани от 0 до $n-1$. В мрежата е *възможно да се стигне от всяка станция до всяка друга станция*, следвайки задължителните посоки на придвижване. Така например, от гарата с номер 0 на фигурата може да се стигне до гарата с номер 2, минавайки през блоковете с номера 2, 3 и 4 по южната линия, обръщайки посоката на движение през станцията с номер 1 в блок 5 и преминавайки през блок 4 по северната линия, стигайки до станция 2 в блок 3.

Тъй като възможностите за придвижване от една гара до друга са многобройни, да дефинираме разстояние между две гари като минималния брой конектори, през които трябва да се премине за да се стигне от едната гара до другата. Например, най-късият в указания смисъл път между станциите 0 и 2 на фигурата е придвижването през блокове 2, 3, 4, 5, 4, 3, което минава през 5 конектора. Затова разстоянието между гарите 0 и 2 е 5.

Движението в мрежата се управлява с компютърна система. За съжаление, след прекъсване на електрическия ток, данните за това какви са блоковете на мрежата и в кои блокове са разположени гарите е изгубена. Единствената информация, която се е съхранила, е номерът на блока в който се намира гарата с номер 0, който винаги е от тип *C*. Освен това, системата все още може да отговаря на въпроси за разстоянието между кои да са две гари. Например, на въпроса какво е разстоянието от гара 0 до гара 2, системата ще отговори с 5.

Задача

Напишете функция `findLocation` която да определя за всяка станция, в кой блок се намира и типа на съответния блок:

- `findLocation(n, first, location, stype)`
  - `n`: брой на станциите.
  - `first`: номерът на блока, в който е гарата с номер 0.
  - `location`: масив с n елемента; в `location[i]` функцията трябва да постави номера на блока, в който се намира гарата с номер i.
  - `stype`: масив с n елемента; в `stype[i]` функцията трябва да постави типа на блока, в който се намира i-тата гара: 1 за блоковете от тип C или 2, за блоковете от тип D

За целта можете да извиквате функцията `getDistance(i, j)`, която връща разстоянието между зададените като аргументи гари `i` и `j`. Извикването `getDistance(i, i)` връща 0. Извикването `getDistance(i, j)` връща -1 ако `i` или `j` е извън интервала $0 \leq i, j \leq n-1$.

Подзадачи

Във всички подзадачи броят m на блоковете не надминава 1000000. В някои подзадачи, броят на извикванията на `getDistance` е ограничен. Ограниченията са различни в различните подзадачи. Програмата ще бъде оценена с 'wrong answer', ако надмине тази граница.

подзадачи	точки	n	извиквания на getDistance	допълнителни изисквания
1	8	$1 \leq n \leq 100$	няма лимит	Всички гари, освен гарата 0, са в блокове от тип *D*.
2	22	$1 \leq n \leq 100$	няма лимит	Всички гари, вдясно от гара 0, са в блокове от тип *D*, а всички гари, вляво от гара 0, са в блокове от тип *C*.
3	26	$1 \leq n \leq 5000$	$n(n-1)/2$	няма други изисквания
4	44	$1 \leq n \leq 5000$	$3(n-1)$	няма други изисквания

Детайли на имплементацията на C/C++

Трябва да изпратите само един файл с име `rail.c` или `rail.cpp`. Той трябва да съдържа имплементация на функцията `findLocation` такава, каквото е описана по-горе, със следната спецификация:

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

като не забравите да включите файла `rail.h`.

Спецификацията на `getDistance` е:

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

Примерен грейдър

Примерният грейдър чете вход във формат:

- ред 1: номер на подзадачата
- ред 2: `n`
- ред $3 + i, 0 \leq i \leq n - 1$: `stype[i]` (1 за тип *C* или 2 за тип *D*), `location[i]`.

Примерният грейдър извежда `Correct`, ако позициите на гарите `location[0]`, `location[1]`, ..., `location[n-1]` и типовете им `stype[0]`, `stype[1]`, ..., `stype [n-1]`, намерени от програмата, съответстват на входните данни, след завършване на работата на `findLocation` или `Incorrect`, когато не съответстват.