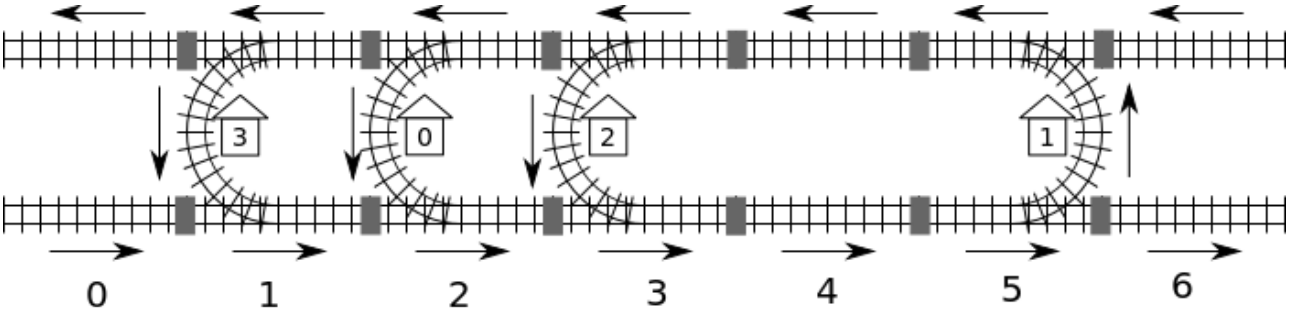
Երկաթգիծ

Թայվանում կա երկաթուղային մեծ ցանց, որը կղզու արևմտյան և արևելյան մասերը իրար է կապում: Երկաթուղին բաղկացած է m բլոկներց: Բլոկները համարակալված են $0, \ldots, m-1$ թվերով, սկսած արևմտյան ափից: Յուրաքանչյուր բլոկի հյուսիսում կա դեպի արևմուտք տանող միակողմանի գիծ, հարավում կա դեպի արևելք տանող միակողմանի գիծ, և նրանց միջև հնարավոր է լինի կայարան:

Կա երեք տիպի բլոկ: *C* տիպի բլոկում կա կայարան, որտեղ պետք է մուտք գործել հյուսիսային գծից և դուրս գալ հարավային գծով: *D* տիպի բլոկում կա կայարան, որտեղ պետք է մուտք գործել հարավային գծից և դուրս գալ հյուսիսային գծով: Կարող է լինել նաև *դատարկ* տիպի բլոկ, որում կայարան չկա: Օրինակ, հետևյալ նկարում 0 բլոկը դատարկ է, 1 բլոկը C տիպի է, իսկ 5 բլոկը D տիպի է: Բլոկները իրար միացված են հորիզոնական ուղղությամբ: Հարևան բլոկների գծերը իրար միացված են *ամրակներով*, որոնք նկարում պատկերված են մուգ ուղղանկյունների տեսքով:



Երկաթուղային համակարգում կա n կայարան: Նրանք համարակալված են 0-ից $n-1$ թվերով: Ենթադրվում է, որ *ցանկացած կայարանից կարելի է գնալ ցանկացած այլ կայարան* երկաթգծերի միջոցով: Օրինակ, 0 կայարանից կարող ենք գնալ 2 կայարանը հետևյալ կերպ: Սկսում ենք 2 բլոկից, հետո անցնում ենք 3 և 4 բլոկների հարավային գծով, ապա անցնում ենք 1 կայարանի միջով, հետո անցնում ենք 4 բլոկի հյուսիսային գծով, և վերջապես 3 բլոկում հասնում ենք 2 կայարան:

Քանի որ հնարավոր են բազմաթիվ երթուղիներ, հեռավորությունը մի կայարանից մինչև մեկ այլ կայարան սահմանվում է որպես այդ երթուղում հանդիպած ամրակների *մինիմալ* քանակ: Օրինակ, 0 կայարանից 2 կայարան տանող կարճագույն երթուղին անցնելով 2-3-4-5-4-3 բլոկներով հատում է 5 ամրակ, հետևաբար, նրա երկարությունը 5 է:

Երկաթուղային ցանցի աշխատանքը կառավարում է համակարգչային համակարգը: Դժբախտաբար, հոսանքազրկումից հետո համակարգիչն այլևս չգիտի, թե որտեղ և ինչ տիպի բլոկներում են գտնվում

կայարանները: Միակ բանը, որ համակարգիչը ստույգ ունի, դա այն է, որ 0 կայարանը միշտ գտնվում է C տիպի բլոկում: Բարեբախտաբար, համակարգիչը կարող է հարցում տալ ցանկացած կայարանից ցանկացած այլ կայարան տանող երթուղու երկարության մասին: Օրինակ, համակարգիչը կարող է հարցնել, թե որքան է 0 կայարանից 2 կայարան տանող երթուղու երկարությունը և, որպես պատասխան, ստանալ 5 թիվը:

Խնդիրը

Պահանջվում է իրականացնել `findLocation` ֆունկցիան, որը յուրաքանչյուր կայարանի համար պարզում է, թե որ համարի բլոկում է այն գտնվում, և ինչ տիպի է այդ բլոկը:

- `findLocation(n, first, location, stype)`
  - `n`: կայարանների քանակը:
  - `first`: 0 կայարանի բլոկի համարը:
  - `location`: n չափի զանգված: Դուք պետք է i-րդ կայարանի բլոկի համարը վերագրեք `location[i]`-ին:
  - `stype`: n չափի զանգված: Դուք պետք է i-րդ կայարանի բլոկի տիպը վերագրեք `stype[i]`-ին: Եթե տիպը C է, պետք է վերագրել 1, եթե տիպը D է, պետք է վերագրել 2:

Կայարանների բլոկների համարները և տիպերը պարզելու համար դուք կարող եք կանչել `getDistance` ֆունկցիան:

- `getDistance(i, j)` վերադարձնում է `i` կայարանից `j` կայարան տանող երթուղու երկարությունը: `getDistance(i, i)` վերադարձնում է 0: `getDistance(i, j)` կվերադարձնի -1, եթե `i`-ն կամ `j`-ն $0 \leq i, j \leq n-1$ տիրույթից դուրս է:

Ենթախնդիրներ

Բոլոր ենթախնդիրներում բլոկների m քանակը չի գերազանցում 1,000,000-ը: Որոշ ենթախնդիրներում սահմանափակում է դրված `getDistance` ֆունկցիայի կանչերի քանակի վրա: Այդ սահմանը տարբեր ենթախնդիրներում տարբեր է: Սահմանը անցնելու դեպքում ձեր ծրագիրը կստանա 'wrong answer':

ենթա-խնդիր	միա-վոր	n	getDistance կանչեր	դիտողություն
1	8	$1 \leq n \leq 100$	անսահմա-նափակ	Բոլոր կայարանները, բացի 0-ից գտնվում են D տիպի բլոկներում:

ենթա-խնդիր	միա-վոր	n	getDistance կանչեր	դիտողություն
2	22	$1 \leq n \leq 100$	անսահմա-նափակ	0 կայարանից դեպի արևելք ընկած բոլոր կայարանները գտնվում են D տիպի բլոկներում, իսկ 0 կայարանից դեպի արևմուտք ընկած բոլոր կայարանները գտնվում են C տիպի բլոկներում:
3	26	$1 \leq n \leq 5,000$	$n(n-1)/2$	լրացուցիչ սահմանափակումներ չկան
4	44	$1 \leq n \leq 5,000$	$3(n-1)$	լրացուցիչ սահմանափակումներ չկան

Իրականացման մանրամասներ

Դուք պետք է հանձնեք ճիշտ մեկ ֆայլ, որի անունը պետք է լինի `rail.c`, `rail.cpp` կամ `rail.pas`: Այդ ֆայլում իրականացրեք `findLocation` ֆունկցիան, ինչպես նկարագրված է վերևում, օգտագործելով `findLocation` ֆունկցիայի ներքևում նշված վերնագիրը: C/C++ ծրագրի դեպքում պետք է նաև ավելացնել `rail.h` ֆայլը ընդգրկելու հրամանը:

C/C++ ծրագիր

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

Pascal ծրագիր

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

`getDistance`-ի վերնագիրը այսպիսին է.

C/C++ ծրագիր

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

Pascal ծրագիր

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

Գրեյդերի օրինակ

Գրեյդերի օրինակը մուտքային տվյալները ներածում է հետևյալ ձևաչափով.

- տող 1: ենթախնդրի համար

- տող 2: `n`
- տող $3+i$, ($0 \le i \le n-1$): `stype[i]` (C տիպի համար 1, D տիպի համար 2), `location[i]`.

Գրեյդերի օրինակը կտպի `Correct` բառը, եթե ձեր ծրագրի հաշված `location[0]` ... `location[n-1]` և `stype[0]` ... `stype[n-1]` արժեքները համապատասխանում են մուտքային տվյալներին։
Չհամապատասխանելու դեպքում կտպվի `Incorrect` բառը։