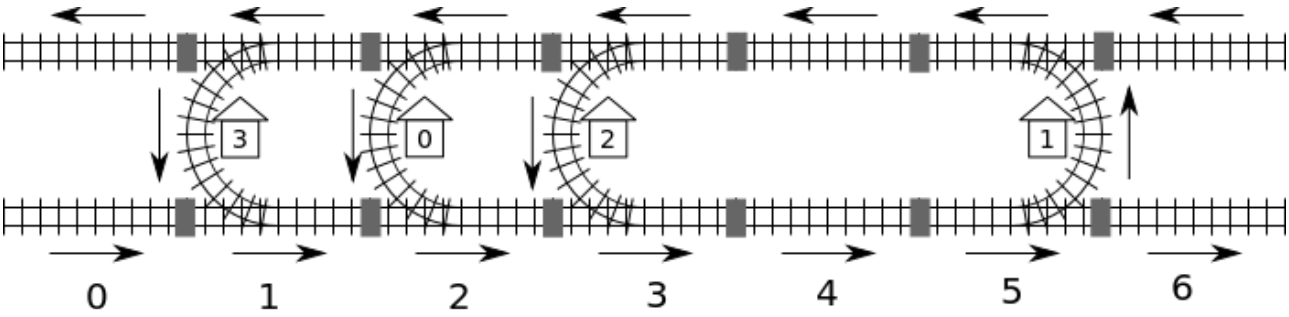
Železnica (Rail)

Hlavná železničná trať na Taiwane spája jeho západné a východné pobrežie. Trať sa skladá z m blokov. Bloky sú zo západu na východ (na obrázkoch teda zľava doprava) očíslované od 0 po $m-1$. Všetky úseky trate sú jednosmerné.

Existujú tri typy blokov: prázdne bloky, bloky typu C a bloky typu D. Každý blok obsahuje dva úseky trate: na severe bloku je úsek idúci na západ a na juhu je úsek idúci na východ. Bloky typov C a D obsahujú aj tretí úsek trate a na ňom stanicu.

Bloky typu C obsahujú trať v tvare písmena C, ktorá vedie zo severnej trate na južnú. Bloky typu D obsahujú trať v tvare oblej časti písmena D, ktorá vedie z južnej trate na severnú. Aj v bloku typu C aj v bloku typu D leží na tomto kuse trate stanica.

Na obrázku nižšie sú bloky 0, 4 a 6 prázdne, bloky 1, 2 a 3 sú typu C a blok 5 je typu D. Medzi jednotlivými blokmi sú na tratiach *spoje*. Tieto sú na obrázku znázornené sivými obdĺžničkami.



V železničnom systéme je dokopy n staníc. Stanice sú očíslované od 0 po $n-1$. Môžete predpokladať, že systém je postavený tak, aby sa z každej stanice dalo dostať na každú inú, cestujúc len povoleným smerom po trati.

V našom príklade na obrázku vieme napríklad cestovať zo stanice 0 na stanicu 2 tak, že začneme v bloku číslo 2 (to je blok, ktorý obsahuje stanicu 0), zídeme na južnú trať, tou prejdeme cez bloky 3 a 4 do bloku 5, tam prejdeme okolo stanice 1 na severnú trať, tou prejdeme cez blok 4 do bloku 3 a tam vôjdeme na stanicu 2.

Vzdialenosť z jednej stanice na druhú si definujeme ako *najmenší možný* počet spojov, cez ktoré po ceste prejdeme. Napríklad zo stanice 0 na stanicu 2 je vyššie popísaná cesta najkratšou možnou cestou. Táto cesta postupne vedie cez bloky 2-3-4-5-4-3, vedie teda cez 5 spojov. V našom príklade je teda vzdialenosť zo stanice 0 na stanicu 2 rovná 5.

Voľakedy sme síce mali mapu železničnej siete, ale poliali sme ju kávou a tak je teraz nečitateľná. Jediné, čo si pamätáme, je, v ktorom bloku leží stanica 0 a že ide o blok typu C. Okrem toho nám už ostal len jeden starý počítač. Ten má celú mapu v pamäti, ale dokáže odpovedať len na jeden typ otázok: môžeš sa ho opýtať na vzdialenosť z ľubovoľnej stanice na ľubovoľnú inú stanicu, a on ti pravdivo odpovie.

Úloha

Tvojou úlohou je napísať funkciu `findLocation` ktorá pre každú stanicu zistí číslo bloku, v ktorom táto stanica leží, a tiež typ dotyčného bloku (C alebo D). Tu sú parametre funkcie `findLocation`:

- `findLocation(n, first, location, stype)`
  - `n`: počet staníc
  - `first`: číslo bloku, v ktorom leží stanica 0
  - `location`: n-prvkové pole, do ktorého máš vyplniť časť odpovede: pre každé i (vrátane 0) na pozíciu i zapíš číslo bloku, v ktorom leží stanica i
  - `stype`: n-prvkové pole, do ktorého máš vyplniť časť odpovede: pre každé i (vrátane 0) na pozíciu i zapíš typ bloku, v ktorom leží stanica i. Presnejšie, zapíš hodnotu 1 pre blok typu C a hodnotu 2 pre blok typu D

Tvoja funkcia `findLocation` môže opakovane volať našu funkciu `getDistance`. Volanie `getDistance(i, j)` ti vráti vzdialenosť zo stanice i na stanicu j. Pre $i = j$ ti `getDistance` vráti 0. Ak použiješ i alebo j pre ktoré neplatí $0 \leq i, j \leq n - 1$, `getDistance` vráti -1.

Podúlohy

V každej podúlohe platí, že počet blokov m neprekročí 1,000,000.

V niektorých podúlohách je počet volaní funkcie `getDistance` obmedzený. Ak toto obmedzenie prekročíš (teda ak sa tvoj program pokúsi `getDistance` zavolať priveľakrát), dostaneš verdikt "`wrong answer`".

podúloha	body	n	max volaní getDistance	poznámky
1	8	$1 \leq n \leq 100$	ľubovoľne veľa	Všetky stanice okrem stanice 0 sú v blokoch typu D.
2	22	$1 \leq n \leq 100$	ľubovoľne veľa	Všetky stanice napravo (na východ) od stanice 0 sú v blokoch typu D. Všetky stanice naľavo (na západ) od stanice 0 sú v blokoch typu C.
3	26	$1 \leq n \leq 5,000$	$n(n-1)/2$	ľubovoľné typy blokov
4	44	$1 \leq n \leq 5,000$	$3(n-1)$	ľubovoľné typy blokov

Detaily implementácie

Odovzdávaš presne jeden súbor, nazvaný `rail.c`, `rail.cpp` alebo `rail.pas`. V tomto súbore by mali byť implementovaná vyššie popísaná funkcia `findLocation`. Musí mať hlavičku uvedenú nižšie. Ak programuješ v C/C++, tvoj súbor musí vložiť (`include`) súbor `rail.h`.

C/C++

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

Pascal

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

Funkcia `getDistance`, implementovaná v našom graderi, má nasledovnú hlavičku:

C/C++

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

Pascal

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

Ukážkový grader

Ukážkový grader, ktorý máte k dispozícii, očakáva vstup v nasledovnom formáte:

- riadok 1: číslo podúlohy
- riadok 2: číslo n
- riadok 3+i pre $0 \leq i \leq n-1$: očakávané hodnoty `stype[i]` (1 pre typ C, 2 pre typ D) a `location[i]`.

Ukážkový grader vypíše "`Correct`" ak váš program správne zistí a vráti všetky hodnoty `location` aj `stype`. V opačnom prípade grader vypíše "`Incorrect`".