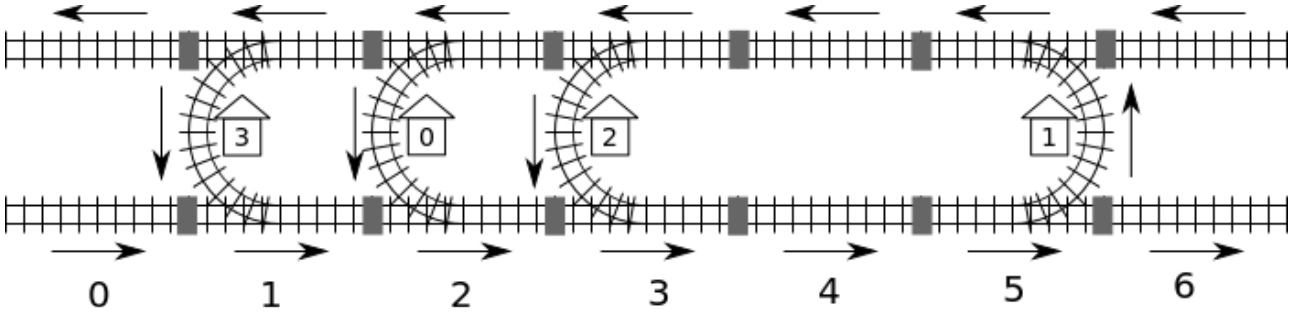
Rail

În Taiwan există o cale ferată lungă care leagă țărmul vestic al insulei de cel estic. Calea ferată este compusă din m blocuri. Blocurile sunt numerotate consecutiv cu numerele $0, \ldots, m-1$ de la vest la est. Fiecare bloc este compus dintr-o linie cu sens unic spre vest situată în nordul blocului, o linie cu sens unic spre est situată în sudul blocului, și opțional o gară între cele două linii.

Există trei tipuri de blocuri. Un bloc de tip *C* are o gară în care trenurile intră de pe linia nordică și din care ies pe linia sudică, un bloc de tip *D* are o gară în care trenurile intră de pe linia sudică și din care ies pe linia nordică, iar un bloc de tip *gol* nu are gară. De exemplu, în figura de mai jos blocurile 0, 4 și 6 sunt de tip *gol*, blocurile 1, 2 și 3 sunt de tip *C*, iar blocul 5 este de tip *D*. Blocurile sunt conectate între ele pe orizontală. Liniile de cale ferată ale două blocuri consecutive sunt legate prin *conectori*, ilustrați prin dreptunghiuri gri în figura următoare.



Pe calea ferată există n gări numerotate de la 0 la $n-1$. Se cunoaște că, folosind calea ferată, *putem ajunge din orice gară în oricare altă gară*. De exemplu putem ajunge din gara 0 în gara 2 plecând din blocul 2, apoi trecând prin blocurile 3 și 4 pe linia sudică, apoi trecând prin gara 1 din blocul 5, apoi trecând prin blocul 4 pe linia nordică și în final ajungând în gara 2 aflată în blocul 3.

Deoarece pot exista mai multe rute posibile, distanța de la o gară la alta se definește ca fiind numărul minim de conectori prin care trece un traseu valid. De exemplu, distanța minimă de la gara 0 la gara 2 este prin blocurile 2-3-4-5-4-3 și trece prin 5 conectori, deci distanța este 5.

Un sistem computerizat monitorizează calea ferată. Din nefericire, după o pană de curent sistemul nu mai cunoaște unde se află gările și în ce tip de bloc se află acestea. Singurul indiciu rămas în sistem este numărul blocului în care se află gara 0, care se află mereu într-un bloc de tip *C*. Din fericire, sistemul poate întreba care este distanța de la orice gară la oricare altă gară. De exemplu, sistemul poate pune următoarea întrebare: 'Care este distanța de la gara 0 la gara 2?', primind ca răspuns 5.

Cerință

Trebuie să implementați funcția `findLocation` care determină pentru fiecare gară numărul și tipul blocului în care se află.

- `findLocation(n, first, location, stype)`
  - `n`: numărul de stații.
  - `first`: numărul blocului care conține gara 0.
  - `location`: un tablou unidimensional de dimensiune n; la finalul execuției acestei funcții numărul blocului în care se află gara i trebuie să se afle în celula `location[i]`.
  - `stype`: un tablou unidimensional de dimensiune n; la finalul execuției acestei funcții tipul blocului în care se află gara i trebuie să se afle în celula `stype[i]`: 1 pentru tipul C sau 2 pentru tipul D.

Puteți apela funcția `getDistance` pentru a vă ajuta să determinați pozițiile și tipurile blocurilor în care se află gările.

- `getDistance(i, j)` returnează distanța de la gara `i` la gara `j`. `getDistance(i, i)` va returna 0. `getDistance(i, j)` va returna -1 dacă `i` sau `j` se află în afara intervalului $0 \leq i, j \leq n-1$.

Subprobleme

Pentru toate subproblemele (subtask-urile) numărul de blocuri m nu depășește 1,000,000. În unele dintre subprobleme numărul de apeluri ale funcției `getDistance` este limitat. Această limită variază de la subproblemă la subproblemă. Programul vostru va primi 'wrong answer' dacă depășește această limită.

subproblemă	puncte	n	apeluri către getDistance	note
1	8	$1 \leq n \leq 100$	nelimitat	Toate gările cu excepția lui 0 sunt în blocuri de tip D.
2	22	$1 \leq n \leq 100$	nelimitat	Toate gările situate în estul gării 0 se află în blocuri de tip D, și toate gările situate la vestul gării 0 se află în blocuri de tip C.
3	26	$1 \leq n \leq 5,000$	$n(n-1)/2$	fără restricții adiționale
4	44	$1 \leq n \leq 5,000$	$3(n-1)$	fără restricții adiționale

Detalii de implementare

Voi trebuie să încărcați exact un fișier, denumit `rail.c`, `rail.cpp` sau `rail.pas`. Acest fișier implementează funcția `findLocation` așa cum este descrisă mai sus, utilizând unul din următoarele antete. Pentru programele C/C++ trebuie să includeți și header-ul `rail.h`.

pentru programele C/C++

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

pentru programele Pascal

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

Antetul funcției `getDistance` este următorul:

pentru programele C/C++

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

pentru programele Pascal

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

Grader-ul de pe computerul vostru

Grader-ul de pe computerul vostru citește datele de intrare în următoarul format:

- linia 1: numărul subproblemei
- linia 2: numărul `n`
- linia $3 + i$, ($0 \leq i \leq n - 1$): numerele `stype[i]` (1 for type C and 2 for type D), și `location[i]`.

Grader-ul de pe computerul vostru va afișă `Correct` dacă `location[0]` ... `location[n-1]` și `stype[0]` ... `stype[n-1]` calculate de programul vostru sunt egale cu valorile corespunzătoare din datele de intrare după ce `findLocation` termină execuția, sau `Incorrect` dacă nu sunt egale.