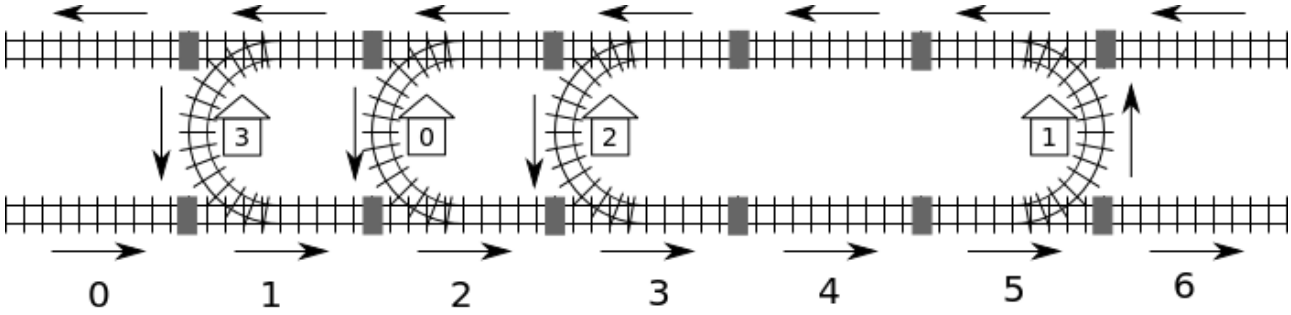
Rail

I Taiwan er det en stor togbane som knytter sammen vestre og østre del av øya. Banen består av m blokker. Blokkene er numerert i rekkefølge $0, \ldots, m-1$, med blokk 0 lengst vest. Hver blokk har et enveis vestgående spor til nord, og et enveis østgående spor til sør. I tillegg kan hver blokk ha én togstasjon mellom disse to sporene.

Det finnes tre typer blokker. En *type C*-blokk har en togstasjon som du må kjøre inn på fra det nordlige sporet og kjøre ut fra til det sørlige sporet. En *type D*-blokk har en togstasjon som du må kjøre inn på fra det sørlige sporet og kjøre ut fra til det nordlige sporet. En *tom* blokk har ikke noen togstasjon. For eksempel, i den følgende figuren er blokk 0, 4 og 6 *tomme*, blokk 1, 2 og 3 er type *C* og blokk 5 er type *D*. Spor i blokker som er ved siden av hverandre er knyttet sammen av *sammenkoblere*, vist som grå rektangler i figuren under.



Togsystemet har n stasjoner nummerert fra 0 til $n-1$. Vi antar at *man kan reise fra en hvilken som helst stasjon til en hvilken som helst annen stasjon* ved å følge togsporene. For eksempel kan man reise fra stasjon 0 til stasjon 2 ved å starte på blokk 2, så passere gjennom blokk 3 og 4 på det sørlige sporet, så passere gjennom stasjon 1 i blokk 5, så passere gjennom blokk 4 på det nordlige sporet, før man når stasjon 2 på blokk 3.

Siden det finnes flere mulige ruter er avstanden fra en stasjon til en annen definert som det *minste* antall sammenkoblere som en rute må passere gjennom. For eksempel går den korteste ruten fra stasjon 0 til stasjon 2 gjennom blokkene 2-3-4-5-4-3; siden den passerer gjennom 5 sammenkoblere, er avstanden 5.

Et datasystem styrer hele togsystemet. Dessverre har det skjedd et strømbrudd og nå vet ikke datamaskinen lenger hvor stasjonene er og hvilke typer blokker de er i. Det eneste datamaskinen vet er hvilken blokk stasjon 0 befinner seg i, og at stasjon 0 er i en type C-blokk. Heldigvis kan datamaskinen spørre om avstanden fra en hvilken som helst stasjon til en hvilken som helst annen stasjon. For eksempel kan den spørre 'Hva er avstanden fra stasjon 0 til stasjon 2?' og få 5 som svar.

Oppgave

Din oppgave er å implementere en funksjon `findLocation` som finner ut blokknummeret og blokktypen for hver stasjon.

- `findLocation(n, first, location, stype)`
  - `n`: antall stasjoner.
  - `first`: blokk nummeret til stasjon 0.
  - `location`: en array av størrelse n; du skal lagre blokk-nummeret til stasjon i i `location[i]`.
  - `stype`: en array av størrelse n; du skal lagre blokk-typen til stasjon i i `stype[i]`: 1 for type C og 2 for type D.

Du kan kalle en ferdiglaget funksjon `getDistance` for å hjelpe deg med å finne posisjonene og typene til stasjonene.

- `getDistance(i, j)` returnerer avstanden fra stasjon`i` til stasjon `j`. `getDistance(i, i)` vil returnere 0. `getDistance(i, j)` vil returnere -1 dersom `i` eller `j` er utenfor intervallet $0 \leq i, j \leq n-1$.

Deloppgaver

I alle deloppgaver er antallet blokker m maksimalt 1,000,000. I noen deloppgaver er det en begrensning på hvor mange ganger du kan kalle `getDistance`. Denne begrensningen varierer mellom deloppgavene. Ditt program vil få 'wrong answer' dersom den overskrider denne begrensningen.

subtask	points	n	getDistance calls	note
1	8	$1 \leq n \leq 100$	ubegrenset	Alle stasjoner untatt 0 er i type D-blokker.
2	22	$1 \leq n \leq 100$	ubegrenset	Alle stasjoner til vest for stasjon 0 er i type D-blokker, og alle stasjoner til øst for stasjon 0 er i type C blokker.
3	26	$1 \leq n \leq 5,000$	$n(n-1)/2$	ingen ytterligere begrensninger
4	44	$1 \leq n \leq 5,000$	$3(n-1)$	ingen ytterligere begrensninger

Implementationsdetaljer

Då må levere inn nøyaktig én fil som heter `rail.c`, `rail.cpp` eller `rail.pas`. Denne filen skal implementere `findLocation` som beskrevet ovenfor med de etterfølgende signaturene. Du må også inkludere header filen `rail.h` for C/C++-implementasjoner.

C/C++-program

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

Pascal-program

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

Signaturen til `getDistance` er som følger:

C/C++ program

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

Pascal program

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

Eksempel-grader

Graderen leser input på følgende format:

- Linje 1: deloppgave-nummeret
- Linje 2: `n`
- Linje $3+i$, ($0 \leq i \leq n-1$): `stype[i]` (1 for type C og 2 for type D), `location[i]`.

Graderen vil skrive ut `Correct` dersom `location[0]` ... `location[n-1]` og `stype[0]` ... `stype[n-1]` som beregnet av programmet ditt matcher inputten når `findLocation` returnerer, eller `Incorrect` dersom de ikke matcher.