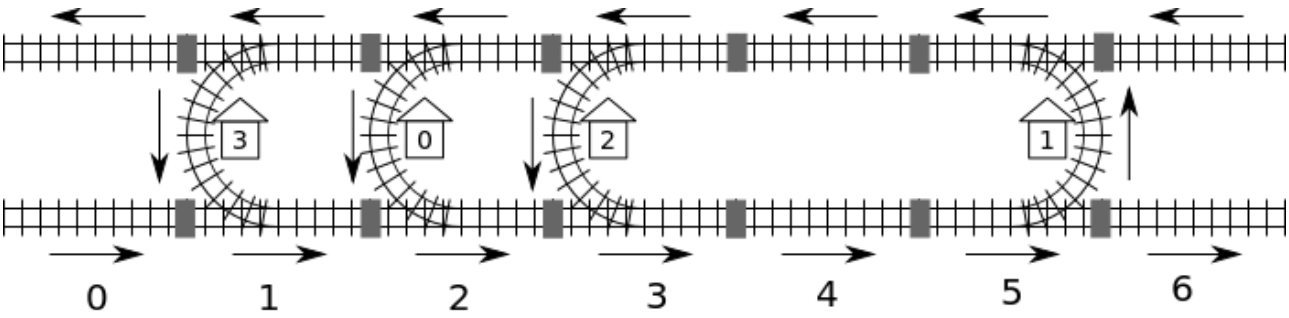
Željeznica

Tajvan ima veliku željezničku liniju koja povezuje zapadnu sa istočnom obalom ostrva. Linija se sastoji od m blokova. Uzastopni blokovi numerisani su $0, \ldots, m-1$, počev sa zapadne strane. Svaki blok ima na sjeveru jednosmjernu stazu usmjerenu na zapad, a na jugu jednosmjernu stazu usmjerenu na istok i, moguće, željezničku stanicu između te dvije staze.

Postoje tri tipa blokova. Blok tipa *C* ima željezničku stanicu u koju se ulazi sa sjeverne staze a izlazi na južnu stazu, dok blok tipa *D* ima željezničku stanicu u koju se ulazi sa južne staze a izlazi na sjevernu stazu. Blok tipa *prazno* nema željezničku stanicu. Na primjer, na sljedećoj slici blokovi 0, 4 i 6 su tipa prazno, blokovi 1, 2 i 3 su tipa C, a blok 5 je tipa D. Staze susjednih blokova spojene su *konektorima*, koji su na slici prikazani kao osjenčeni pravougaonici.



Železnički sistem ima n stanica koje su numerisane od 0 do $n-1$. Pretpostavljamo da je *moguće stići iz bilo koje stanice u bilo koju drugu* krećući se po stazama. Na primjer, možemo stići iz stanice 0 do stanice 2 tako što krenemo iz bloka 2, zatim prođemo kroz blokove 3 i 4 južnom stazom, onda prođemo kroz stanicu 1 u bloku 5, pa kroz blok 4 sjevernom stazom i stižemo u stanicu 2 u bloku 3.

Kako ima više mogućih puteva, rastojanje između dvije stanice definiše se kao *minimalni* broj konektora kroz koje put prolazi. Na primjer, najkraći put od stanice 0 do stanice 2 je kroz blokove 2-3-4-5-4-3 i prolazi kroz 5 konektora, što znači da je rastojanje 5.

Kompjuterski sistem upravlja željeznicom. Nažalost, nakon nestanka struje kompjuter više ne zna gdje se stanice nalaze niti u kojem su tipu bloka. Jedini trag koji kompjuter ima je broj bloka u kome se nalazi stanica 0, a koji je uvijek blok tipa C. Srećom, kompjuter može pronaći rastojanje između bilo koje dvije stanice. Na primjer, on može tražiti odgovor na pitanje 'koliko je rastojanje od stanice 0 do stanice 2?' i dobiti 5.

Zadatak

Neophodno je da implementirate funkciju `findLocation` koja za svaku stanicu određuje broj i tip bloka.

- `findLocation(n, first, location, stype)`

- `n`: broj stanica.
- `first`: broj bloka stanice 0.
- `location`: niz veličine n; trebate smjestiti broj bloka stanice i u `location[i]`.
- `stype`: niz veličine n; trebate smjestiti tip bloka stanice i u `stype[i]`: 1 za tip C a 2 za tip D.

Možete pozvati funkciju `getDistance` da Vam pomogne u nalaženju lokacija i tipova stanica.

- `getDistance(i, j)` vraća rastojanje od stanice `i` do stanice `j`. `getDistance(i, i)` će vratiti 0. `getDistance(i, j)` će vratiti -1 ako je `i` ili `j` van opsega $0 \leq i, j \leq n-1$.

Podzadaci

U svim podzadacima broj blokova m nije veći od 1,000,000. U nekim podzadacima broj poziva funkcije `getDistance` je ograničen. Ograničenje je drugačije za različite podzadatke. Vaš program će dobiti 'wrong answer' (pogrešan odgovor) ako prekorači ograničenje.

podzadatak	poeni	n	broj poziva getDistance	komentar
1	8	$1 \leq n \leq 100$	neograničeno	Sve stanice osim 0 su u blokovima tipa D.
2	22	$1 \leq n \leq 100$	neograničeno	Sve stanice koje su istočno od stanice 0 su u blokovima tipa D, a sve stanice zapadno od stanice 0 su u blokovima tipa C.
3	26	$1 \leq n \leq 5,000$	$n(n-1)/2$	nema dodatnih ograničenja
4	44	$1 \leq n \leq 5,000$	$3(n-1)$	nema dodatnih ograničenja

Implementacioni detalji

Morate predati tačno jedan fajl, sa imenom `rail.c`, `rail.cpp` ili `rail.pas`. Ovaj fajl implementira funkciju `findLocation` koristeći sljedeću signaturu (potpis, zaglavlje). Dodatno, neophodno je da uključite (include) heder fajl `rail.h` za C/C++ program.

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);
```

Signatura (potpis, zaglavlje) funkcije `getDistance` je kao što slijedi.

C/C++ program

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

Pascal program

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

Ocjenjivač

Ocjenjivač čita ulaz u sljedećem formatu:

- linija 1: broj podzadatka
- linija 2: `n`
- linije $3 + i$, ($0 \leq i \leq n - 1$): `stype[i]` (1 za tip C a 2 za tip D), `location[i]`.

Ocjenjivač će odštampati `Correct` (ispravno) ako se vrijednosti `location[0]` ... `location[n-1]` i `stype[0]` ... `stype[n-1]` koje je vaš program izračunao poklapaju sa ulazom ocjenjivača, ili `Incorrect` ako to nije slučaj.