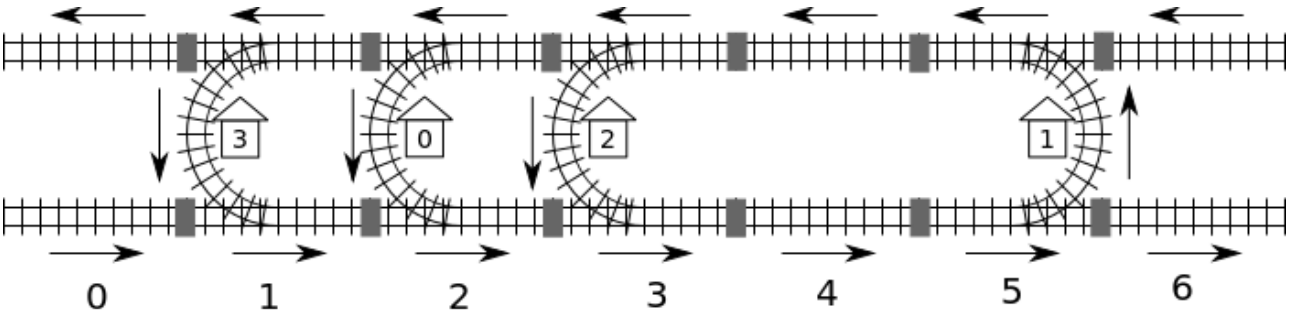
Σιδηρόδρομος

Η Taiwan έχει μια μεγάλη σιδηροδρομική γραμμή που συνδέει τη δυτική και την ανατολική ακτή του νησιού. Η γραμμή αποτελείται από m τμήματα, αριθμημένα στο διάστημα $0, \ldots, m-1$, αρχίζοντας από το δυτικό άκρο. Κάθε τμήμα αποτελείται από μία γραμμή στο βόρειο μέρος του με κατεύθυνση προς τα δυτικά, μία γραμμή στο νότιο μέρος του με κατεύθυνση προς τα ανατολικά, και προαιρετικά ένα σταθμό μεταξύ αυτών των δύο.

Υπάρχουν τρεις τύποι τμημάτων. Ένα τμήμα τύπου *C* έχει ένα σταθμό στον οποίο η είσοδος είναι από τη βόρεια γραμμή και η έξοδος είναι στη νότια γραμμή. Ένα τμήμα τύπου *D* έχει ένα σταθμό στον οποίο η είσοδος είναι από τη νότια γραμμή και η έξοδος είναι στη βόρεια γραμμή. Τέλος, ένα *κενό* τμήμα δεν έχει σταθμό. Για παράδειγμα, στο παρακάτω σχήμα τα τμήματα 0, 4 και 6 είναι κενά, τα τμήματα 1, 2 και 3 είναι τύπου C, και το τμήμα 5 είναι τύπου D. Τα τμήματα συνδέονται μεταξύ τους οριζόντια. Οι γραμμές των τμημάτων ενώνονται με *συνδέσεις*, που φαίνονται στο σχήμα σαν γκρίζα ορθογώνια.



Υπάρχουν συνολικά n σταθμοί αριθμημένοι στο διάστημα $0, \ldots, n-1$. Υποθέτουμε ότι *μπορούμε να πάμε από οποιονδήποτε σταθμό σε οποιονδήποτε άλλο σταθμό* ακολουθώντας τις σιδηροδρομικές γραμμές. Για παράδειγμα, μπορούμε να πάμε από το σταθμό 0 στο σταθμό 2 ξεκινώντας από το τμήμα 2, ύστερα περνώντας από τα τμήματα 3 και 4 στη νότια γραμμή, περνώντας από το τμήμα 5 μέσα από το σταθμό 1, ύστερα περνώντας από το τμήμα 4 στη βόρεια γραμμή, και τέλος φθάνοντας στο σταθμό 2 στο τμήμα 3.

Καθώς υπάρχουν πολλές δυνατές διαδρομές, η απόσταση από ένα σταθμό σε έναν άλλο ορίζεται ως το *ελάχιστο* πλήθος συνδέσεων από τις οποίες περνά η διαδρομή. Για παράδειγμα, η συντομότερη διαδρομή από το σταθμό 0 στο σταθμό 2 είναι μέσω των τμημάτων 2-3-4-5-4-3 και περνά από 5 συνδέσεις, άρα η απόσταση είναι 5.

Ένας υπολογιστής ελέγχει το σιδηροδρομικό δίκτυο. Δυστυχώς, μετά από μία διακοπή ρεύματος, ο υπολογιστής δεν ξέρει πια πού βρίσκονται οι σταθμοί και σε τι τύπου τμήματα. Το μόνο στοιχείο που έχει είναι ο αριθμός του τμήματος όπου βρίσκεται ο σταθμός 0, ο οποίος βρίσκεται πάντοτε σε τμήμα τύπου C. Ευτυχώς, ο υπολογιστής μπορεί να ρωτήσει την απόσταση από οποιονδήποτε σταθμό σε οποιονδήποτε άλλο σταθμό. Για παράδειγμα, ο υπολογιστής μπορεί να ρωτήσει 'ποια είναι η απόσταση από το σταθμό 0 στο σταθμό 2;' και θα λάβει την απάντηση 5.

Πρόβλημα

Πρέπει να υλοποιήσετε τη συνάρτηση `findLocation` που βρίσκει για κάθε σταθμό σε ποιο τμήμα βρίσκεται και τι τύπου είναι.

- `findLocation(n, first, location, stype)`
  - `n`: το πλήθος των σταθμών.
  - `first`: ο αριθμός του τμήματος όπου βρίσκεται ο σταθμός 0.
  - `location`: πίνακας μεγέθους n. Στη θέση `location[i]` πρέπει να βάλετε τον αριθμό του τμήματος όπου βρίσκεται ο σταθμός i.
  - `stype`: πίνακας μεγέθους n. Στη θέση `stype[i]` πρέπει να βάλετε τον τύπο του τμήματος όπου βρίσκεται ο σταθμός i: 1 για τύπο C και 2 για τύπο D.

Μπορείτε να καλείτε τη συνάρτηση `getDistance` για να βοηθηθείτε να βρείτε τις θέσεις και τους τύπους των σταθμών.

- Η κλήση `getDistance(i, j)` επιστρέφει την απόσταση από το σταθμό `i` στο σταθμό `j`. Η κλήση `getDistance(i, i)` θα επιστρέψει 0. Η κλήση `getDistance(i, j)` θα επιστρέψει -1 αν το `i` ή το `j` είναι εκτός του διαστήματος $0 \leq i, j \leq n-1$.

Υποπροβλήματα

Σε όλα τα υποπροβλήματα, το πλήθος των τμημάτων m δε θα υπερβαίνει το 1,000,000. Σε κάποια υποπροβλήματα, το πλήθος των κλήσεων στη συνάρτηση `getDistance` είναι περιορισμένο. Το όριο διαφέρει ανάλογα με το υποπρόβλημα. Οι υποβολές σας θα τερματιστούν με 'wrong answer' αν υπερβούν αυτό το όριο.

υποπρόβλημα	βαθμοί	n	κλήσεις στη getDistance	σημείωση
1	8	$1 \leq n \leq 100$	απεριόριστες	Όλοι οι σταθμοί εκτός από τον 0 είναι σε τμήματα τύπου D.
2	22	$1 \leq n \leq 100$	απεριόριστες	Όλοι οι σταθμοί στα ανατολικά του σταθμού 0 είναι σε τμήματα τύπου D, και όλοι οι σταθμοί στα δυτικά του σταθμού 0 είναι σε τμήματα τύπου C.
3	26	$1 \leq n \leq 5,000$	$n(n-1)/2$	κανένας επιπλέον περιορισμός
4	44	$1 \leq n \leq 5,000$	$3(n-1)$	κανένας επιπλέον περιορισμός

Λεπτομέρειες υλοποίησης

Πρέπει να υποβάλετε ακριβώς ένα αρχείο, με όνομα `rail.c`, `rail.cpp` ή `rail.pas`. Αυτό το αρχείο πρέπει να υλοποιεί το υποπρόγραμμα `findLocation` όπως περιγράφεται παραπάνω, το οποίο πρέπει να έχει μία από τις παρακάτω επικεφαλίδες. Για τις υλοποιήσεις σε C/C++, το αρχείο σας πρέπει να κάνει include το αρχείο επικεφαλίδας `rail.h`.

Πρόγραμμα C/C++

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

Πρόγραμμα Pascal

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

Η επικεφαλίδα της `getDistance` είναι ως εξής.

Πρόγραμμα C/C++

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

Πρόγραμμα Pascal

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

Ενδεικτικός βαθμολογητής

Ο ενδεικτικός βαθμολογητής διαβάζει την είσοδο με την ακόλουθη μορφή:

- γραμμή 1: ο αριθμός του υποπροβλήματος
- γραμμή 2: `n`
- γραμμή $3 + i$, $(0 \leq i \leq n - 1)$: `stype[i]` (1 για τύπο C και 2 για τύπο D), `location[i]`.

Ο ενδεικτικός βαθμολογητής θα τυπώσει `Correct` αν τα `location[0]` ... `location[n-1]` και `stype[0]` ... `stype[n-1]` που υπολογίζονται από το πρόγραμμά σας ταιριάζουν με την είσοδο, όταν η συνάρτηση `findLocation` επιστρέφει, διαφορετικά θα τυπώσει `Incorrect`.