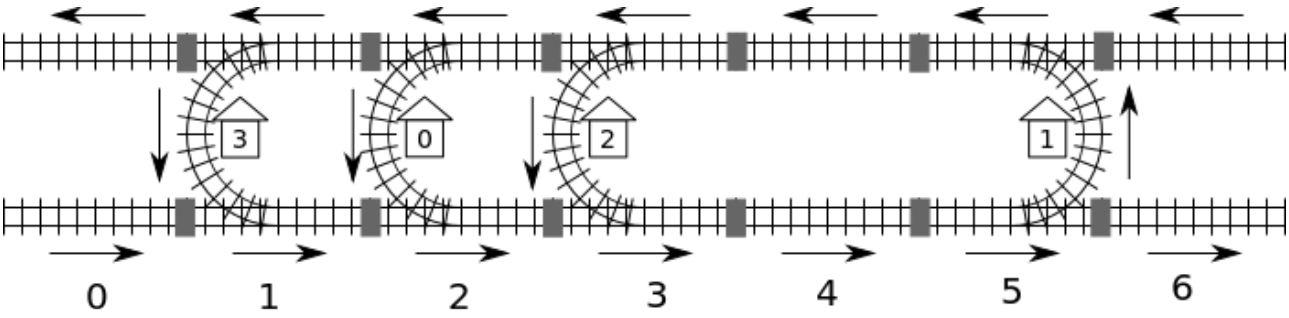
Rail

Taiwan possiede una vasta linea ferroviaria che collega la costa occidentale a quella orientale dell'isola. La linea consiste di m blocchi consecutivi, numerati come $0, \ldots, m-1$ a partire dalla costa occidentale. Ciascun blocco è composto da un binario superiore da percorrere a senso unico da est verso ovest, da un binario inferiore da percorrere a senso unico da ovest verso est e, opzionalmente, da una stazione ferroviaria collocata tra i due binari.

Vi sono tre tipi di blocchi. Un blocco di tipo *C* ha la stazione ferroviaria che prevede l'ingresso dal binario superiore e l'uscita verso il binario inferiore; un blocco di tipo *D* ha la stazione ferroviaria che prevede l'ingresso dal binario inferiore e l'uscita verso il binario superiore; un blocco di tipo *empty* non contiene alcuna stazione ferroviaria. Per esempio, nella figura seguente, il blocchi 0, 4 e 6 sono di tipo empty, i blocchi 1, 2 e 3 sono di tipo C, e il blocco 5 è di tipo D. I blocchi sono connessi orizzontalmente e i binari di due blocchi adiacenti sono collegati tramite *giunzioni*, mostrate come rettangoli pieni in figura.



La linea ferroviaria ha n stazioni numerate da 0 a $n-1$. È sempre possibile *andare da ogni stazione a ogni altra* seguendo i binari. Per esempio, si può andare dalla stazione 0 alla stazione 2 partendo dal blocco 2, passando attraverso i blocchi 3 e 4 lungo il binario inferiore, quindi passando attraverso la stazione 1 nel blocco 5, poi attraverso il blocco 4 con il binario superiore, e finalmente raggiungendo la stazione 2 nel blocco 3.

Essendoci diversi possibili percorsi, la distanza da una stazione all'altra è definita come il *minimo* numero di giunzioni attraversate da un percorso. Per esempio, il percorso più breve dalla stazione 0 alla stazione 2 attraversa i blocchi 2-3-4-5-4-3 e passa attraverso 5 giunzioni, per cui la distanza risulta essere 5.

Un sistema automatico gestisce la linea ferroviaria mediante un computer. Sfortunatamente, dopo una caduta di tensione, il computer ha perso i dati e quindi non sa il tipo dei blocchi e in quali di essi sono collocate le stazioni: l'unica traccia rimasta è che il computer conosce il numero del blocco che contiene la stazione 0, che è sempre collocata in un blocco di tipo C. Fortunatamente il computer può formulare delle query che restituiscono la distanza da una qualunque stazione a una qualunque altra stazione. Per esempio, il computer può formulare la query 'qual è la distanza dalla stazione 0 alla stazione 2?' e ricevere 5 come risposta.

Descrizione del problema

Devi implementare una funzione `findLocation` che determini per ciascuna stazione il numero e il tipo del blocco che la contiene.

- `findLocation(n, first, location, stype)`
  - `n`: numero delle stazioni.
  - `first`: numero del blocco contenente la stazione 0.
  - `location`: array di lunghezza n; devi scrivere il numero del blocco contenente la stazione i in `location[i]`.
  - `stype`: array di lunghezza n; devi scrivere il tipo del blocco contenente la stazione i in `stype[i]` utilizzando 1 per indicare il tipo C e 2 per il tipo D.

Puoi chiamare una funzione `getDistance` per trovare il numero e il tipo dei blocchi contenenti le stazioni.

- `getDistance(i, j)` restituisce la distanza dalla stazione `i` alla `j`. `getDistance(i, i)` restituisce 0. `getDistance(i, j)` restituisce -1 se `i` o `j` sono fuori dell'intervallo ammissibile $0 \le i, j \le n - 1$.

Subtask

In tutti i subtask il numero di blocchi m è al massimo $1\,000\,000$. In alcuni subtask il numero di chiamate a `getDistance` è limitato. I limiti variano a seconda del subtask. Il tuo programma riceverà 'wrong answer' se supera questo limite.

subtask	punti	n	chiamate a getDistance	note
1	8	$1 \le n \le 100$	illimitate	Tutte le stazioni tranne la 0 sono in blocchi di tipo D.
2	22	$1 \le n \le 100$	illimitate	Tutte le stazioni a est della stazione 0 sono in blocchi di tipo D, e tutte le stazioni a ovest della stazione 0 sono in blocchi di tipo C.
3	26	$1 \le n \le 5\,000$	$n(n-1)/2$	nessun limite aggiuntivo
4	44	$1 \le n \le 5\,000$	$3(n-1)$	nessun limite aggiuntivo

Dettagli di implementazione

Devi sottoporre esattamente un file, di nome `rail.c`, `rail.cpp` o `rail.pas`. Il file deve implementare le procedure descritte sopra utilizzando le intestazioni seguenti. Per i programmi in C/C++, devi anche includere il file header `rail.h`.

Linguaggio C/C++

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

Linguaggio Pascal

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

L'intestazione di `getDistance` è come segue.

Linguaggio C/C++

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

Linguaggio Pascal

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

Grader di esempio

Il grader di esempio legge input nel formato seguente:

- riga 1: numero del subtask
- riga 2: `n`
- righe $3 + i$, ($0 \leq i \leq n - 1$): `stype[i]` (1 per il tipo C e 2 per il tipo D), `location[i]`.

Il grader di esempio stamperà `Correct` se i valori di `location[0]` ... `location[n-1]` e `stype[0]` ... `stype[n-1]` calcolati dal vostro programma corrispondono a quelli dati in input quando la procedura `findLocation` termina, o `Incorrect` se non corrispondono.