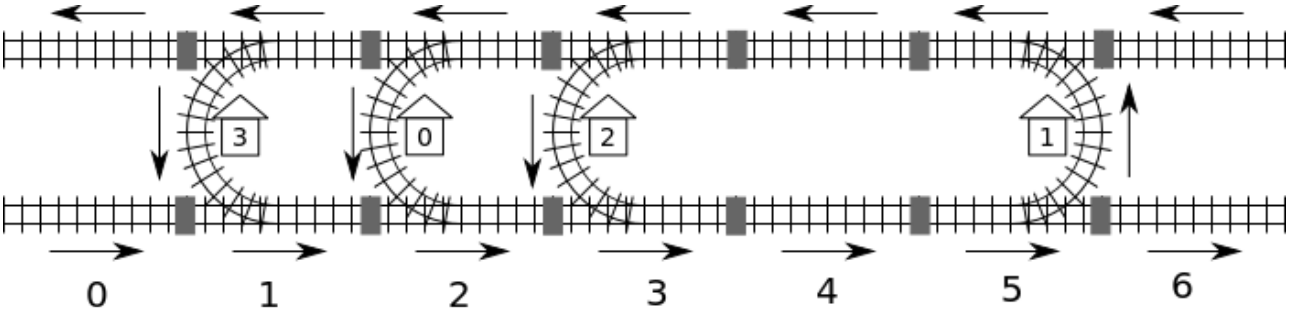
Rail

Taïwan a une importante ligne ferroviaire connectant l'île d'est en ouest. La ligne est composée de m blocs. Les blocs consécutifs sont numérotés $0, \ldots, m-1$ en partant de l'extrémité ouest. Chaque bloc est composé d'une voie nord dont le trafic va uniquement d'est en ouest, d'une voie sud dont le trafic va d'ouest en est, et éventuellement d'une gare entre ces deux voies.

Il existe trois types de blocs: Un bloc de type *C* possède une gare dont l'entrée se fait via la voie nord et la sortie via la voie sud. Un bloc de type *D* possède une gare dont l'entrée se fait via la voie sud et la sortie via la voie nord. Un bloc *vide* ne possède pas de gare. Par exemple, dans le schéma suivant, les blocs 0, 4 et 6 sont de type vide, les blocs 1, 2 et 3 sont de type C, et le bloc 5 est de type D. Les blocs sont connectés les uns aux autres horizontalement. Les voies de blocs adjacents sont reliées via des *jonctions*, représentées par des rectangles foncés sur le schéma.



Le système ferroviaire possède n gares numérotées de 0 à $n-1$. On vous garantit qu'*on peut se rendre à toutes les gares à partir de n'importe quelle gare* en suivant les voies. Par exemple, on peut aller de la gare 0 à la gare 2 en démarrant du bloc 2, en traversant les blocs 3 et 4 de la voie sud, en passant à côté de la gare 1 située dans le bloc 5, en traversant le bloc 4 de la voie nord, pour finalement arriver à la gare 2 située dans le bloc 3.

Vu qu'il existe plusieurs itinéraires possibles, la distance d'une gare à l'autre est définie comme le nombre *minimal* de jonctions traversées. Par exemple, l'itinéraire le plus court de la gare 0 à la gare 2 passe par les blocs 2-3-4-5-4-3 et traverse 5 jonctions, la distance est donc de 5.

Le système ferroviaire est géré par un système informatique. Malheureusement, suite à une coupure de courant, celui-ci ne connait plus la position des gares ainsi que le type des blocs où elles se trouvent. L'unique information qu'il possède toujours est le numéro du bloc de la gare 0 qui est toujours située sur un bloc de type C. Heureusement, le système informatique peut toujours demander la distance d'une gare à une autre. Par exemple, il peut obtenir la réponse à 'quelle est la distance entre la gare 0 et la gare 2 ?' et recevra 5.

Tâche

Vous devez implémenter la fonction `findLocation` qui détermine, pour chaque gare, le numéro ainsi que le type du bloc sur lequel elle se trouve.

- `findLocation(n, first, location, stype)`
  - `n`: le nombre de gares.
  - `first`: le numéro du bloc de la gare 0.
  - `location`: un tableau de taille n; vous devez placer le numéro du bloc de la gare i dans `location[i]`.
  - `stype`: un tableau de taille n; vous devez placer le type du bloc de la gare i dans `stype[i]`: 1 pour le type C et 2 pour le type D.

Vous pouvez appeler la fonction `getDistance` afin de déterminer les emplacements et les types de gare.

- `getDistance(i, j)` retourne la distance entre la gare `i` et la gare `j`. `getDistance(i, i)` retournera 0. `getDistance(i, j)` retournera -1 si `i` ou `j` est hors des bornes $0 \leq i, j \leq n-1$.

Sous-tâches

Dans chaque sous-tâche, le nombre de blocs m ne sera pas supérieur à 1 000 000. Dans certaines sous-tâches, le nombre d'appels à `getDistance` est limité. La limite est différente selon la sous-tâche. Votre programme recevra 'wrong answer' s'il dépasse cette limite.

sous-tâche	points	n	appels à getDistance	note
1	8	$1 \leq n \leq 100$	illimités	Toutes les gares sauf la gare 0 sont sur des blocs de type D.
2	22	$1 \leq n \leq 100$	illimités	Toutes les gares à l'est de la gare 0 sont de type D, toutes les gares à l'ouest de la gare 0 sont de type C.
3	26	$1 \leq n \leq 5000$	$n(n-1)/2$	aucune limite supplémentaire
4	44	$1 \leq n \leq 5000$	$3(n-1)$	aucune limite supplémentaire

Détails d'implémentation

Vous devez soumettre un seul fichier, appelé `rail.c`, `rail.cpp` ou `rail.pas`. Ce fichier implémente la fonction `findLocation` telle que décrite précédemment en utilisant les signatures suivantes. Pour le C/C++, vous devez inclure (`#include`) le fichier d'en-tête `rail.h`.

Programme C/C++

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

Programme Pascal

```
function findLocation(n, first: longint, var location, stype: array of lo
```

Les signatures de la fonction getDistance sont les suivantes :

Programme C/C++

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

Programme Pascal

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

Évaluateur

L'évaluateur fourni lit l'entrée dans le format suivant:

- ligne 1: le numéro de sous-tâche
- ligne 2: `n`
- ligne $3 + i$, ($0 \leq i \leq n - 1$): `stype[i]` (1 pour le type C et 2 le type D), `location[i]`.

L'évaluateur affichera `Correct` si `location[0]` ... `location[n-1]` et `stype[0]` ... `stype[n-1]` calculés par votre programme correspondent à l'entrée fournie ou `Incorrect` si elle ne correspond pas.