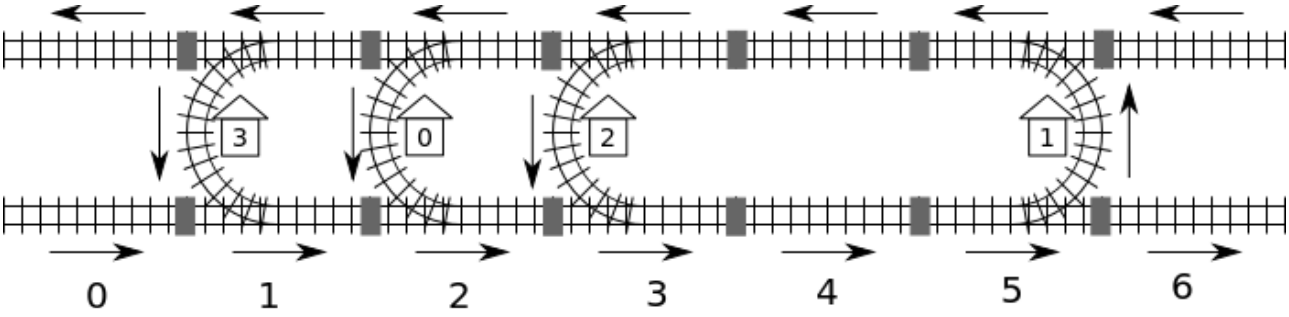
Rail

Taiwan tiene una gran línea de ferrocarril conectando las costas oeste y este de la isla. La línea consta de m bloques. Los bloques consecutivos están numerados $0, \ldots, m-1$, partiendo del extremo occidental. Cada bloque tiene un tendido viniendo del este en su norte, y un tendido viniendo del oeste al sur, y opcionalmente una estación de tren entre ellas.

Hay tres tipos de bloques. Un bloque de tipo *C* tiene una estación de tren al que debes entrar desde el tendido norte y salir hacia el tendido sur, un bloque de tipo *D* tiene una estación de tren al cual debes entrar desde el tendido sur y salir hacia l] tendido norte, y un bloque de tipo *vacío* no tiene estación de tren. Por ejemplo, en la figura siguiente el bloque 0 es de tipo vacío, el bloque 1 es de tipo C, y el bloque 5 es de tipo D. Los bloques se conectan uno con otro horizontalmente. Tendidos de bloques adyacentes son unidos por *conectores*, mostrados como rectángulos sombreados en la figura siguiente.



El sistema de vías tiene n estaciones numeradas desde 0 hasta $n-1$. Suponemos que *podemos ir desde cualquier estación hacia otra estación cualquiera* siguiendo el tendido. Por ejemplo, podemos ir desde la estación 0 hacia la estación 2 partiendo desde el bloque 2, después pasando por los bloques 3 y 4 por el tendido sur, y después por la estación 1, después por el bloque 4 por el tendido norte, yfinalmente alcanzando la estación 2 en el bloque 3.

Como hay múltiple rutas posibles, la distancia desde una estación hacia otra está definida como el *mínimo* número de conectores por los cuales la ruta pasa. Por ejemplo, la ruta desde estación 0 hacia 2 toca los bloques 2-3-4-5-4-3 y pasa a través de 5 conectores, luego, la distancia es 5.

Un sistema de cómputo administra el sistema de rieles. Desfortunadamente después de una caída de tensión la computadora no recuerda donde están las estaciones ni en que tipos de bloques están. La única pista es que la computadora tiene el número de bloque de la estación 0, que está siempre en un bloque de tipo C. Afortunadamente la computadora puede interrogar la distancia desde cualquier estación hacia una otra. Por ejemplo, la computadora puede interrogar '¿cuál es la distancia desde la estación 0 hacia la estación 2?' y recibirá 5.

Tareas

Debes implantar una función `findLocation` que determine para cada estación el número del bloque y su tipo.

- `findLocation(n, first, location, stype)`
  - `n`: el número de estaciones.
  - `first`: el número del bloque de la estación 0.
  - `location`: array de tamaño n; debes colocar el número del bloque de estación i into `location[i]`.
  - `stipo`: array de tamaño n; debes colocar el tipo del bloque dela estación i en `stipo[i]`: 1 para tipo C y 2 para tipo D.

Puedes llamar una función `getDistancia` para ayudarte a encontrarlas ubicaciones ytipos de estaciones.

- `getDistancia(i, j)` devuelve la distancia desde la estación `i` hasta la estación `j`. `getDistancia(i, i)` devolverá 0. `getDistancia(i, j)` devolverá -1 si `i` o `j` están fuera del rango $0 \le i, j \le n - 1$.

Subtareas

En todas las subtareas el número de bloques m no supera 1.000.000. En algunas subtareas el número de llamadas a `getDistancia` está limitado. Este límite varía según la subtarea. Tu programa recibirá 'wrong answer' si excede el líimite.

subtarea	points	n	llamadas a getDistance	note
1	8	$1 \le n \le 100$	ilimitado	Todas las estaciones excepto la 0 están en bloques de tipo D.
2	22	$1 \le n \le 100$	ilimitado	Todas las estaciones al este de la 0 están en bloques de tipo D y todas las estaciones al oeste de la 0 están en bloques de tipo C.
3	26	$1 \le n \le 5,000$	$n(n-1)/2$	sin límites adicionales
4	44	$1 \le n \le 5,000$	$3(n-1)$	sin límites adicionales

Detalles de implantación

Debes enviar exactamente un archivo, llamado `rail.c`, `rail.cpp` or `rail.pas`. Este archivo implementa la `findLocation` tal como está descripta arriba usando los siguientes encabezamientos. Debes incluir también una archivo de encabezamiento `rail.h` para las implantaciones en C/C++.

programa en C/C++

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

programa en Pascal

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

Los encabezmientos de `getDistancia` son como sigue.

program a C/C++

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

programa en Pascal

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

Sample grader

El sample grader (programa para probar ejemplos propios) lee la entrada en el siguiente formato:

- línea 1: el número de subtarea
- línea 2: `n`
- línea $3 + i$, ($0 \leq i \leq n - 1$): `stipo[i]` (1 para tipo C y 2 para tipo D), `location[i]`.

El sample grader imprimirá `Correct` si `location[0]` ... `location[n-1]` y`stipo[0]` ... `stipo[n-1]` computados por tu programa coincide con la entrada cuando vuelve de `findLocation`, or `Incorrect` si no coincide.