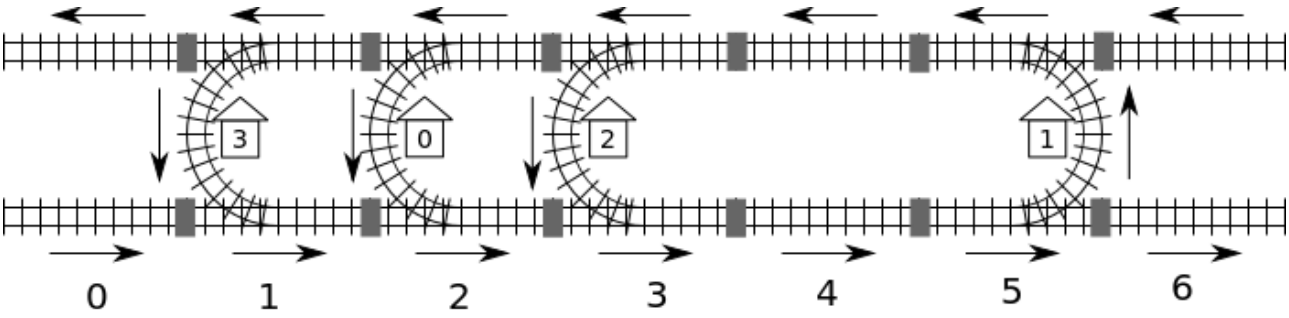
# Gleis

Taiwan hat eine lange Gleisverbindung von der West- zur Ostküste der Insel. Diese Verbindung besteht aus $m$ Abschnitten, die im Westen beginnend von $0$ bis $m-1$ durchnummeriert sind. Jeder Abschnitt hat im Norden ein Einweg-Gleis von Osten nach Westen und im Süden ein Einweg-Gleis von Westen nach Osten. Zusätzlich haben manche Abschnitte einen Bahnhof zwischen beiden Gleisen.

Es gibt drei Typen von Abschnitten. Ein Abschnitt vom Typ *C* enthält einen Bahnhof, den man vom nördlichen Gleis befahren und dann zum südlichen Gleis hin verlassen muss. Ein Abschnitt vom Typ *D* enthält einen Bahnhof, den man vom südlichen Gleis befahren und dann zum nördlichen Gleis hin verlassen muss. Ein Abschnitt vom Typ *leer* enthält keinen Bahnhof. Beispielsweise sind in der folgenden Abbildung die Abschnitte 0, 4 und 6 leer, die Abschnitte 1, 2 und 3 vom Typ C, und Abschnitt 5 vom Typ D. Benachbarte Abschnitte sind über *Verbindungsstücke* (schattierte Rechtecke) miteinander verbunden.




Das Schienennetz hat $n$ Bahnhöfe, nummeriert von $0$ bis $n-1$. Du kannst davon ausgehen, dass *jeder Bahnhof von jedem anderen aus zu erreichen* ist. Zum Beispiel kann man von Bahnhof 0 zu Bahnhof 2 gelangen. Dazu kann man von Abschnitt 2 auf dem südlichen Gleis Abschnitte 3 und 4 durchfahren, dann durch Bahnhof 1 im Abschnitt 5 fahren, danach auf dem nördlichen Gleis Abschnitt 4 durchfahren, und erreicht schließlich Bahnhof 2 im Abschnitt 3.

Da es mehrere Routen geben kann, ist der Abstand von einem Bahnhof zu einem anderen definiert als die *minimale* Anzahl an Verbindungsstücken (schattierte Rechtecke), durch die man fahren muss. Zum Beispiel verläuft die kürzeste Route von Bahnhof 0 zu Bahnhof 2 durch die Abschnitte 2-3-4-5-4-3 und enthält 5 Verbindungsstücke, d.h. der Abstand ist 5.

Das Schienennetz wird von einem Computer verwaltet. Leider weiß der Computer nach einem Stromausfall nicht mehr, wo die Bahnhöfe sind und in welchen Typen von Abschnitten sich diese befinden. Der einzige Hinweis, den der Computer noch hat, ist die Nummer des Abschnittes, der Bahnhof 0 enthält, und dass dieser den Typ C hat. Zum Glück kann der Computer den Abstand zwischen zwei beliebigen Bahnhöfen abfragen. Beispielsweise kann er "Was ist der Abstand von Bahnhof 0 zu Bahnhof 2?" fragen und wird als Antwort 5 erhalten.

# Task

Deine Aufgabe ist es, eine Funktion `findLocation` zu schreiben, die für jeden Bahnhof ermittelt, in welchem Abschnitt er liegt und von welchem Typ dieser Abschnitt ist.

- `findLocation(n, first, location, stype)`
  - `n`: die Anzahl der Bahnhöfe.
  - `first`: die Nummer des Abschnitts, in dem sich Bahnhof 0 befindet.
  - `location`: Array der Größe $n$; die Funktion soll die Abschnittsnummer von Bahnhof $i$ in `location[i]` schreiben.
  - `stype`: Array der Größe $n$; die Funktion soll den Typ von Bahnhof $i$ in `stype[i]` schreiben: 1 für Typ C und 2 für Typ D.

Du kannst die Funktion `getDistance` aufrufen um den Abstand zwischen zwei Bahnhöfen abzufragen.

- `getDistance(i, j)` gibt den Abstand von Bahnhof `i` nach Bahnhof `j` zurück. `getDistance(i, i)` gibt 0 zurück. `getDistance(i, j)` gibt -1 zurück, falls `i` oder `j` außerhalb des Intervalls $0 \leq i, j \leq n-1$ liegt.

# Subtasks

In allen Subtasks ist die Anzahl $m$ der Abschnitte höchstens 1.000.000. In einigen Subtasks ist die Anzahl der Aufrufe von `getDistance` beschränkt. Das Limit variiert je nach Subtask. Du bekommst die Antwort 'wrong answer', falls dein Programm dieses Limit überschreitet.

| Subtask | Punkte | $n$ | Aufrufe von getDistance | Bemerkung |
|---|---|---|---|---|
| 1 | 8 | $1 \leq n \leq 100$ | unbeschränkt | Alle Bahnhöfe außer 0 sind vom Typ D. |
| 2 | 22 | $1 \leq n \leq 100$ | unbeschränkt | Alle Bahnhöfe östlich von 0 sind vom Typ D und alle Bahnhöfe westlich von 0 sind vom Typ C. |
| 3 | 26 | $1 \leq n \leq 5.000$ | $n(n-1)/2$ | Keine weiteren Einschränkungen. |
| 4 | 44 | $1 \leq n \leq 5.000$ | $3(n-1)$ | Keine weiteren Einschränkungen. |

# Implementierungsdetails

Du sollt genau eine Datei einsenden mit dem Namen `rail.c`, `rail.cpp` oder `rail.pas`. Diese Datei soll die Funktion `findLocation` wie oben beschrieben mit einer der folgenden Signaturen implementieren. Für C/C++ musst du außerdem die Headerdatei `rail.h` einbinden.

## Programme in C/C++

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

## Programme in Pascal

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

Die Signatur von `getDistance` ist wie folgt.

## Programme in C/C++

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

## Programme in Pascal

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

## Sample-Grader

Der Sample-Grader liest die Eingabe in folgendem Format:

- Zeile $1$: Die Nummer des Subtasks.
- Zeile $2$: `n`
- Zeile $3 + i$, ($0 \leq i \leq n - 1$): `stype[i]` (1 für Typ C und 2 für Typ D), `location[i]`.

Der Sample-Grader gibt `Correct` aus, falls die von deinem Programm berechneten Werte `location[0]` ... `location[n-1]` und `stype[0]` ... `stype[n-1]` mit der Eingabe übereinstimmen, und `Incorrect`, falls sie abweichen.