Esteras numéricas

Sobre un riel de números enteros se coloca una maquinaria. Dicha maquinaria se mueve usando una estera formada por números. Para poder ubicarse correctamente, los números de la parte de la estera que se apoya sobre el riel deben coincidir con los números del riel y no puede haber parte de la estera fuera del mismo (como muestra en las figuras 1 y 2).

Figura 1. Maquinaria puesta sobre rieles de manera correcta.

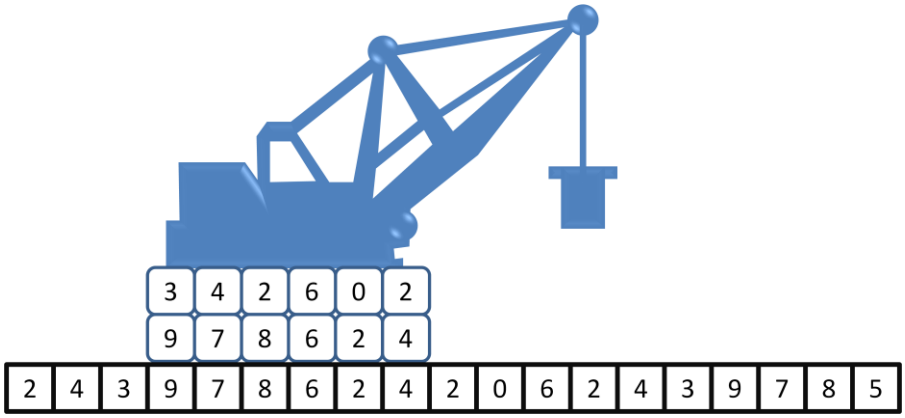
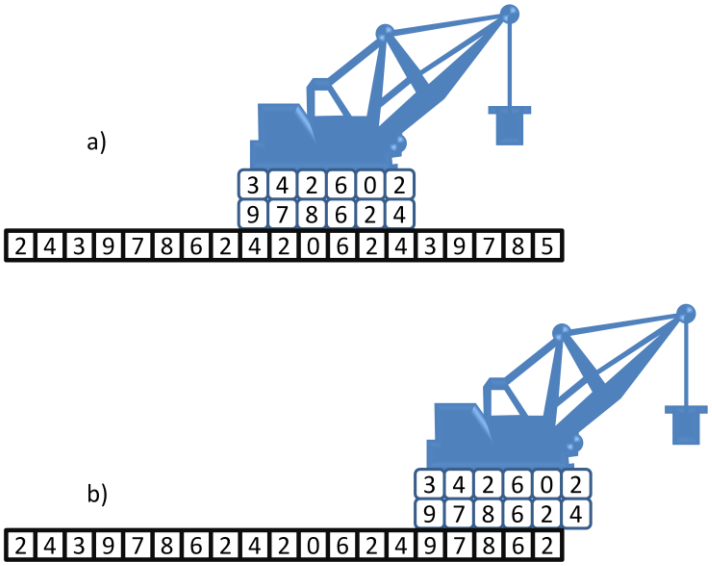
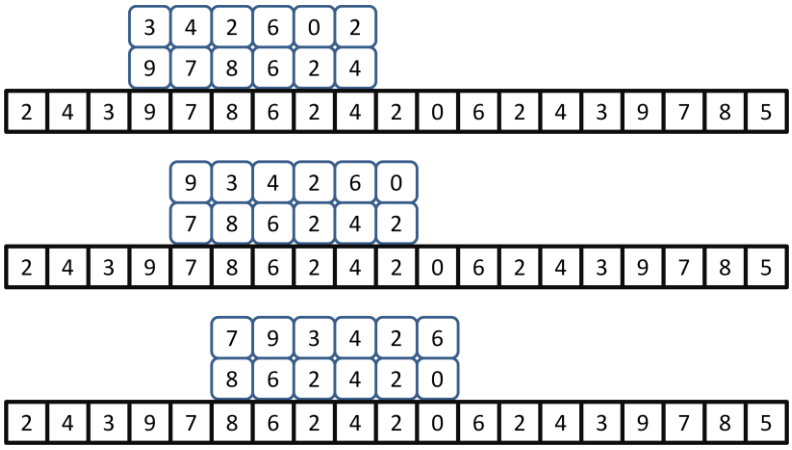


Figura 2. Maquinaria ubicada sobre rieles de manera errónea a) los números no coinciden b) hay parte de la estera fuera del riel.



En cada movimiento que la maquinaria realiza hacia el frente la estera sufre una rotación en sentido de las manecillas del reloj. Para poder avanzar, los números de la nueva base de la maquinaria deben coincidir nuevamente con los números del riel y no puede salirse del riel (Figura 3).

Figura 3. Estera avanzando 2 pasos.



Usted deberá implementar un algoritmo que determine el mayor número de pasos hacia delante que puede realizar la maquinaria sobre determinado riel.

La información de la estera se recibe en forma de **array** bidimensional de `int` (`int[,]`) de 2 filas (dimensión 0) y cualquier cantidad de columnas (dimensión 1). La primera fila del **array** representa la parte superior de la estera y la segunda fila la parte inferior. El riel se recibe en forma de **array** de `int` (`int[]`). La posición inicial de la maquinaria se recibe en un tercer parámetro de tipo `int`.

Usted deberá implementar una biblioteca `ExamenEsteras` con un método dentro de la clase `Rieles` con nombre `Avance`.

```
namespace ExamenEsteras
{
    public class Rieles
    {
        public static int CantidadDePasos(int[,] esteras, int[] riel, int posicion)
        {
            // TODO: Implemente aquí su algoritmo
        }
    }
}
```

El método recibe la información sobre la estera, el riel y la posición inicial de la maquinaria. Debe devolver un entero con la cantidad máxima de pasos que pudo dar la maquinaria hacia delante. Su método deberá lanzar la excepción `InvalidArgumentException` en caso que la maquinaria en un inicio esté ubicada incorrectamente.

Ejemplo. Para el caso que se muestra en la figura 1, su método será llamado en la forma:

```
Rieles.CantidadDePasos(
    new int[,]
        {
            { 3, 4, 2, 6, 0, 2 },
            { 9, 7, 8, 6, 2, 4 }
        },
    new int[] { 2, 4, 3, 9, 7, 8, 6, 2, 4, 2, 0, 6, 2, 4, 3, 9, 7, 8, 5 },
    3);
```

Y deberá devolver 9.