Grupo Weboo
Universidad de La Habana

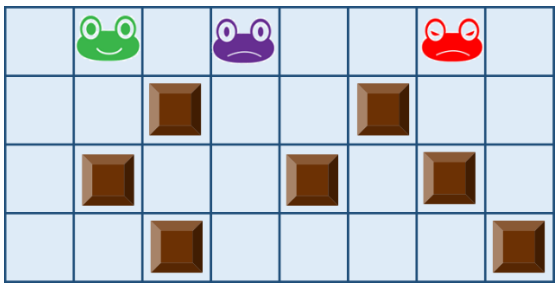
Segundo Examen de Programación

Curso 2018-2019

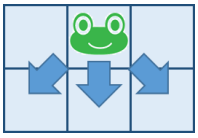
Ranas Suizas

NOTA: Si usted está leyendo este documento sin haber extraído el compactado que se le entregó, ciérrelo ahora, extraiga todos los archivos en el escritorio, y siga trabajando desde ahí. Es un error común trabajar en la solución dentro del compactado, lo cual provoca que los cambios no se guarden. Si usted comete este error y entrega una solución vacía, no tendrá oportunidad de reclamar.

En un charco rectangular de N filas y M columnas se ubican en un inicio K ranas en la fila 0. En otras posiciones del lago pueden aparecer chocolates. Todas las ranas salen de la primera fila al mismo tiempo. La figura muestra un ejemplo de una posible distribución.

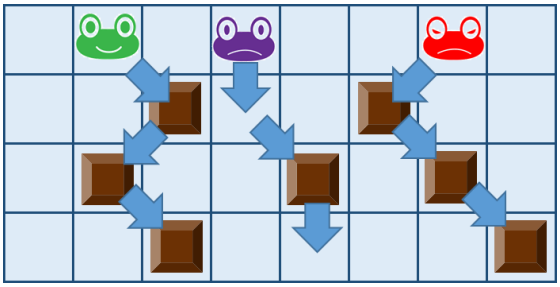


En cada paso las ranas avanzan a alguna de las tres casillas debajo de ellas.



Si una rana ocupa la posición de un chocolate se lo come. No pueden coincidir dos ranas en la misma posición. A las ranas les gusta mucho el chocolate y quieren comer la mayor cantidad de chocolates posibles pero no saben cómo. Lo que cuenta es que **el total de chocolate** que se puedan comer sea el **máximo**, aun cuando para ello haya alguna rana deba "sacrificarse".

En el ejemplo de la figura inicial un óptimo puede ser el siguiente:



Su tarea es implementar un algoritmo que dada una configuración inicial del charco, determine la mayor cantidad de chocolates que las ranas pueden comerse.

Usted debe haber recibido junto a este documento una solución de Visual Studio con dos proyectos: una biblioteca de clases (*Class Library*) y una aplicación de consola (*Console Application*). Usted debe implementar el método `ComiendoChocolates` en el namespace Weboo.Examen que está dentro de la clase `Charco`.

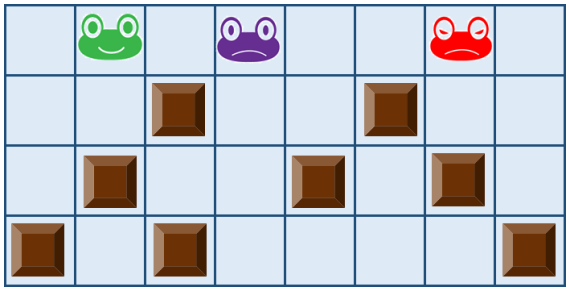
El método `ComiendoChocolates` recibe un array bidimensional de `bool` que indica en cada posición si hay chocolate o no, y un array con K enteros indicando las columnas iniciales desde donde parten las ranas. Este array está ordenado y bien formado, es decir, no hay columnas repetidas ni con valores fuera del rango de columnas del charco. Su método debe devolver un entero con el máximo número de chocolates que el grupo de ranas podrían comerse en el trayecto de la primera fila a la última.

NOTA: Todo el código de la solución debe estar en este proyecto (biblioteca de clases), pues es el único código que será evaluado. Usted puede adicionar todo el código que considere necesario, pero no puede cambiar los nombres del namespace, clase o método mostrados. De lo contrario, el probador automático fallará.

Ejemplos

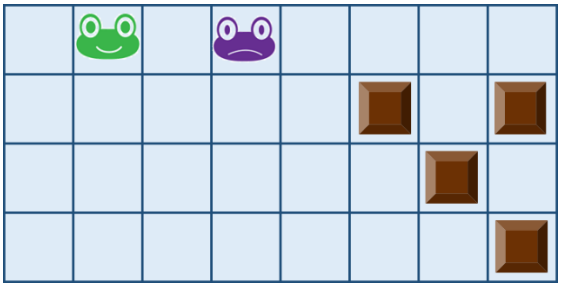
```
int resultado1 = Charco.ComiendoChocolates(new bool[,] {
    { false, false, false, false, false, false, false, false },
    { false, false, true,  false, false, true,  false, false },
    { false, true,  false, false, true,  false, true,  false },
    { false, false, true,  false, false, false, false, true  }
}, new int[] { 1, 3, 6 });
// Máxima cantidad de chocolates que pueden comer: 7
```

Note que es posible que las ranas no alcancen todos los chocolates.



```
int resultado2 = Charco.ComiendoChocolates(new bool[,] {
    { false, false, false, false, false, false, false, false },
    { false, false, true,  false, false, true,  false, false },
    { false, true,  false, false, true,  false, true,  false },
    { true,  false, true,  false, false, false, false, true  }
}, new int[] { 1, 3, 6 });
// Máxima cantidad de chocolates que pueden comer: 7
```

Incluso puede haber casos en los que no se alcance chocolate alguno.



```
int resultado3 = Charco.ComiendoChocolates(new bool[,] {
    { false, false, false, false, false, false, false, false },
    { false, false, false, false, false, true,  false, true  },
    { false, false, false, false, false, false, true,  false },
    { false, false, false, false, false, false, false, true  }
}, new int[] { 1, 3 });
// Máxima cantidad de chocolates que pueden comer: 0
```

NOTA: Los casos de prueba que aparecen en este proyecto son solamente de ejemplo. Que usted obtenga resultados correctos con estos casos no es garantía de que su solución sea correcta y de buenos resultados con otros ejemplos. De modo que usted debe probar con todos los casos que considere convenientes para comprobar la validez de su implementación.