Sub-palíndromo mayor

NOTA: *Si está leyendo este documento directamente del archivo* ExamenProgramacion.zip*, ciérrelo ahora, descompacte todo el contenido en su escritorio, y vuelva a abrir este documento desde allí.*

En analogía con los strings, un *array* se dice palíndromo si tiene los mismos valores recorridos de izquierda a derecha que derecha a izquierda. Por ejemplo, los siguientes *arrays* son palíndromos:

```
int[] a = { 4, 2, 10, 9, 9, 10, 2, 4 };

int[] b = { 66, 2, 8,  2, 66 };

int[] c = { 2 };
```

Un *subarray* de *i* a *j* de un *array* `a` es el segmento del *array* que incluye los elementos de `a[i]` hasta `a[j]`. Por ejemplo, a continuación se muestra un *array* y posibles *subarrays* del mismo.

```
int[] array = { 10, 9, 9, 10, 2, 10, 9, 7 };

int[] sub1 = { 10, 9, 9, 10 }; // Sub-array de 0 a 3.

int[] sub2 = { 9, 9 }; // Sub-array de 1 a 2.

int[] sub3 = { 10, 2, 10 }; // Sub-array de 3 a 5.

int[] sub4 = { 9, 10, 2, 10, 9 }; // Sub-array de 2 a 6.
```

El problema a implementar consiste en a partir de un *array* **devolver la longitud del mayor *subarray* palíndromo que puede estar contenido dentro de éste**.

Observe que si el propio *array* es un palíndromo la longitud del mayor *subarray* palíndromo es la del propio *array*, por otra parte todo *array* no vacío tiene al menos un *subarray* palíndromo ya que un *sub-array* de un solo elemento es un palíndromo.

Note que no importa si hay más de un *subarray* con igual longitud mayor porque no hay que dar el *subarray* en la respuesta sino solo la longitud.

Junto a este documento debe haber descargado una solución de Visual Studio con 2 proyectos. El proyecto *Weboo.ExamenProgramacion* consiste en una biblioteca de clases, donde encontrará el siguiente fragmento de código:

```
public static int MayorSubpalindromo(int[] array)
{

}
```

En el proyecto *Weboo.ExamenProgramacion.Probador* (aplicación de consola) encontrará algunos ejemplos de prueba. Por supuesto que el hecho de que su método de un resultado correcto para estos ejemplos no garantiza que su solución sea correcta, pruebe con tantos ejemplos como considere necesario.

Recuerde que su respuesta será evaluada de forma automática invocando al método `Resolver` y comprobará el valor que retorne este método con la respuesta correcta. Por este motivo usted no debe cambiar ni el nombre ni los parámetros del método ni tampoco el nombre de la clase `MayorPalindromo`. Puede añadir otros métodos si lo considera útil.

Usted debe entregar el archivo *Weboo.ExamenProgramacion.dll* y **todos** los archivos de código fuente (extensión *.cs*) necesarios para compilar correctamente su código, en un compactado *.zip*. Por este motivo le sugerimos escribir todo el código de su respuesta en un único archivo *MayorPalindromo.cs* para simplificar este proceso. Usted no necesita entregar el código adicional que haya programado para hacer pruebas.

Nota:

- Los *arrays* que se pasarán como parámetro para hacer la prueba automática nunca serán `null` y siempre tendrán longitud mayor que cero.
- Recuerde compilar y guardar a menudo. Así evitará pérdida de información por falla de hardware o de electricidad.

Ejemplos

A continuación le suministramos algunos ejemplos, que encontrará además en el proyecto de prueba.

```
int[] array1 = { 1, 2, 3, 4, 5, 4, 3, 2, 1 };
int solucion1 = 9;
// porque el propio array es un palíndromo.

int[] array2 = { 1, 2, 1, 4, 2, 2, 2, 2, 10 };
int solucion2 = 4;
// el subarray { 1, 2, 1 } es palíndromo de longitud 3 pero
// el subarray { 2, 2, 2, 2 } también lo es y tiene longitud 4.

int[] array3 = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
int solucion3 = 1;
// cualquier elemento hace de subarray palíndromo,
// no hay ninguno de mayor longitud.

int[] array4 = { 10, 4, 3, 15, 4, 3, 11, 3, 4, 15, 100 };
int solucion4 = 7;
// { 4, 3, 15, 4, 3 } es palíndromo pero
// también lo es { 15, 4, 3, 11, 3, 4, 15 }.
```