We call an element of an array its *peak* if the array is strictly increasing on the left side of the *peak* and strictly decreasing on its right side. So if we have an array called sequence of length n, sequence[k] is a *peak* if and only if

sequence[0] < sequence[1] < ... < sequence[k]

and

sequence[k] > sequence[k + 1] > ... > sequence[n - 1]

Knowing that sequence definitely has a *peak* and that the *peak* is not its first or last element, find the *peak's* value.

**Example**

For sequence = [1, 2, 4, 7, 9, 3, -2, -10], the output should be  
sequencePeakElement(sequence) = 9.

**Input/Output**

* **[execution time limit] 3 seconds (cs)**
* **[input] array.integer sequence**

*Guaranteed constraints:*  
3 ≤ sequence.length ≤ 10,  
-100 ≤ sequence[i] ≤ 100.

* **[output] integer**

**[C#] Syntax Tips**

// Prints help message to the console

// Returns a string

string helloWorld(string name) {

Console.Write("This prints to the console when you Run Tests");

return "Hello, " + name;

}

<https://codefights.com/challenge/GMkfrHjyKvf2yjhy8/solutions>

static int sequencePeakElement(int[] sequence)

{

return sequence.Max();

}