**Finding an array in an array**

Try to find not one element in the array, but a whole subarray!

If the subarray is found in the array, then you need to return the minimum index from which the subarray begins in the original array. For example, searching for the subarray "3,4" in the array "1,2,3,4,3,4" should return 2.

It can be written more strictly if we denote array as array, and subarray: the function must return such a minimum k that array [k + i] == subarray [i] for all i from 0 to subarray.Length-1.

If the subarray is not found, then -1 should be returned.

Consider that an empty subarray is contained in any array starting at index 0.

We helped you and wrote FindSubarrayStartIndex, leaving only one ContainsAtIndex helper method unimplemented:

public static int FindSubarrayStartIndex(int[] array, int[] subArray)

{

for (var i = 0; i < array.Length - subArray.Length + 1; i++)

if (ContainsAtIndex(array, subArray, i))

return i;

return -1;

}

Your task is to implement the ContainsAtIndex method that it uses.

**Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace umop5o5zFindingAnArrayInAnArray

{

class Program

{

static void Main(string[] args)

{

int[] array = { 1, 2, 3, 4, 3, 4 };

int[] subArray = { 3, 4 };

Console.WriteLine(FindSubarrayStartIndex(array, subArray));

Console.ReadKey();

}

public static int FindSubarrayStartIndex(int[] array, int[] subArray)

{

for (var i = 0; i < array.Length - subArray.Length + 1; i++)

if (ContainsAtIndex(array, subArray, i))

return i;

return -1;

}

public static bool ContainsAtIndex(int[] array, int[] subArray, int i)

{

bool real=true;

for (int j = 0; j < subArray.Length; j++)

{

if (array[j + i] == subArray[j])

real = true;

else

{

real = false; break;

}

}

return real;

}

}

}