**Joining arrays**

Implement a Combine method that returns an array composed of the supplied arrays.

To create a new array, use the static Array.CreateInstance method, which takes the type of the array element.

To find out the type of elements in a passed array, use myArray.GetType (). GetElementType ().

Check that the element types are the same in all passed arrays!

If the resulting array cannot be created, return null.

public static void Main()

{

var ints = new[] { 1, 2 };

var strings = new[] { "A", "B" };

Print(Combine(ints, ints));

Print(Combine(ints, ints, ints));

Print(Combine(ints));

Print(Combine());

Print(Combine(strings, strings));

Print(Combine(ints, strings));

}

static void Print(Array array)

{

if (array == null)

{

Console.WriteLine("null");

return;

}

for (int i = 0; i < array.Length; i++)

Console.Write("{0} ", array.GetValue(i));

Console.WriteLine();

}

**Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace umop13o8zJoiningArrays

{

class Program

{

public static void Main()

{

var ints = new[] { 1, 2 };

var strings = new[] { "A", "B" };

Print(Combine(ints, ints));

Print(Combine(ints, ints, ints));

Print(Combine(ints));

Print(Combine());

Print(Combine(strings, strings));

Print(Combine(ints, strings));

Console.ReadKey();

}

static void Print(Array array)

{

if (array == null)

{

Console.WriteLine("null");

return;

}

for (int i = 0; i < array.Length; i++)

Console.Write("{0} ", array.GetValue(i));

Console.WriteLine();

}

public static Array Combine (params Array[] array)

{

if (array.Length == 0)

return null;

var type = array[0].GetType().GetElementType();

var length = 0;

foreach (var ar in array)

{

if (ar.GetType().GetElementType() != type)

return null;

length+=ar.Length;

}

var result = Array.CreateInstance(type, length);

var index = 0;

foreach (var ar in array)

foreach (var el in ar)

result.SetValue(el, index++);

return result;

}

}

}