# Returning an array from a function

**The goal for this exercise** is to make sure that you return an array from a method (as a return value). Also: to give you more practice creating and using methods/functions.

**What you need to do for this exercise:**

1. In the starter project, add code to the Returning\_An\_Array class, so that the RunExercise method does the following:
	1. Declare an integer array variable, but DO NOT ALLOCATE THE ARRAY ITSELF.
	2. Call the **CreateRandomlyFilledArray** method (found in the ArrayReturnMethods class) which will allocate (and then return, via the return value) a brand-new array. This method will take as it's only parameter an integer, which tells the method how long (how many elements) the array should have. The array is then created inside the method, filled with values that have been randomly created by the method. The random values should vary between 0 and 100 (inclusive of both 0 and 100).

		1. You may assume that the parameter’s value is a positive number. In other words, you are NOT required to check for zero, or negative numbers.
			1. As always, you’re welcome to do this check, it’s just not required ☺
	3. The array will then be passed (as a parameter) to the **PrintArray** method (found in the ArrayReturnMethods class), which will take as it's single parameter an array of integers, and will print out everything in the array.

		1. If you haven’t implemented the PrintArray method yet, you should. It’s pretty simple, good practice, and very useful.
		2. It should take as it’s only parameter an array of integers, and it should print out (using Console.WriteLine) every element of the array.
	4. There is a block of commented-out code for the Returning\_An\_Array class, including some ‘sample code’ that helps to demonstrate how to use the methods that you’ll be writing. You need to uncomment this, and build your answer on it.
	Some of this code you will have to modify to complete this exercise, and the rest of it you are allowed to modify (but don't modify the code to the point where it avoids the goal of this exercise ☺ )