Task 3.2P Answer Sheet

Name: Le Ba Tung
Student ID: 104175915

1. In 2.2P, how many Counter objects were created?

There are 2 Counter objects explicitly create.
Counter 1 and Counter 2

2. Variables declared without the "new" keyword are different to the objects created when we call "new". Referring to the main method in task 2.2P, what is the relationship between the variables initialised with and without the "new" keyword?

The relationship between the variables initialize with and without the "new" keyword is that 'myCounters[0]' and 'myCounters[1]' refer to distinct 'Counter' objects create using "new" , while 'myCounter[2]' refers to the same 'Counter' object as 'myCounters[0]'. Any changes made to 'myCounters[0]' will also affect 'myCounter[2]' , as they both point to the same object.

3. In 2.2P, explain why resetting the counter in myCounters[2] also changed the value of the counter in myCounters[0].

Resetting the counter in myCounters[2] also changes the value of the counter in myCounters[0] because both myCounters[0] and myCounters[2] point to the same counter object in memory. When you reset it through one reference, the change is reflected in the other reference because they share the same underlying object.

4. The key difference between memory on the heap and memory on the stack is that the heap holds "dynamically allocated memory". What does this mean? In your answer, focus on the size and lifetime of the allocations.

Heap memory is like a storage area where you can put things of different sizes as you need them. These things can stay there for a long time until you decide to take them out.

Stack memory is like a stack of plates. You can only put plates of a fixed size on top, and when you're done with a plate, you remove it immediately.

So, the key difference is that heap memory is more flexible in terms of size and how long things can stay there, while stack memory is more rigid with fixed sizes and things are removed as soon as you're done with them.

5. Are objects allocated on the heap or the stack? What about local variables?

Objects: Think of objects as things you create, like a toy or a book. These are usually kept in a special storage area called the "heap." You can create as many objects as you want, and they can stick around for a long time.

Local Variables: Local variables are like notes or sticky notes you use for temporary information. These are typically placed on a stack, which is like a pile of notes. You use them for quick tasks, and when you're done, you remove them immediately. They don't last very long.

6. What does the new() method do when called for a particular class, and what does it return?

When you use new to create an object from a class, it does two main things:
Allocates Space: It sets aside a special place in the computer's memory to store all the information that belongs to the object.
Gets it Ready: It runs a special function called a constructor to set up the object with its initial values and behaviors.
Then, it gives you back a special code (a reference) that you can use to find and work with that object later. So, new both creates an object and gives you the way to interact with it.

7. Assuming the class Counter exists in my project, if I wrote the code "Counter myCounter;" (note there is no "="), what value would myCounter have? Why?

Certainly! When you write Counter myCounter; without the = sign, it means you're declaring a variable named myCounter, but you haven't given it a value yet. In C#, if you don't give a value to a variable, it gets a default value.
For reference types like classes (such as Counter), the default value is null. Think of null as a special value that means "no value" or "empty." So, in this case, myCounter will have a value of null, which means it doesn't point to any actual object. To use myCounter, you would need to assign an instance of the Counter class to it using the new keyword or some other method.

8. Based on the code you wrote in task 2.2P, draw a diagram showing the locations of the variables and objects in main and their relationships to one another.

Stack	Heap
args (string[]): Command line arguments passed to Main method. **Main** PrintCounters: A static method within the Counter namespace that takes an array of Counter objects as a parameter. - MainClass: A class within the Counter namespace containing the Main method. - Console: System.Console class used for writing output to the console.	- myCounters (Counter[]): An array of Counter objects. - Index 0: Counter object named "Counter 1" - Index 1: Counter object named "Counter 2" - Index 2: Reference to the same Counter object as Index 0 ("Counter 1")