# Task 3.2P Answer Sheet

Name: Nguyen Huy Anh

Student ID: 104180281

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

In 2.2P, three Counter objects were created: myCounter[0], myCounter[1], and myCounter[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?

Variables declared without the "new" keyword refer to existing objects, but objects generated with the "new" keyword are newly instantiated. In task 2.2P's main method, the variable myCounter[2] refers to the same object as myCounter[0] because it was assigned myCounter[0]'s reference via the phrase myCounter[2] = myCounter[0].

1. 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 modified the value of the counter in myCounters[0], because both myCounters[0] and myCounters[2] link to the same Counter object in memory. They effectively point to the same memory region, so any changes done with one reference affect the object retrieved with the other.

## 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.

The heap stores dynamically allocated memory, which is allocated at runtime and can be allocated and deallocated as needed. This memory is not limited by the scope of functions or blocks, and it can be accessed worldwide. Objects allocated on the heap have a longer lifetime and their size is not determined at compile time.

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

Objects are allocated on the heap, while local variables are usually allocated on the stack. Local variables have a limited lifetime and are deallocated when they leave scope, which is often at the end of the block in which they are defined.

1. What does the new() method do when called for a particular class, and what does it return?  
   In C#, the new() function creates a new instance of a class. When invoked for a certain class, it allocates memory for a new object of that class, sets its fields to their default values, and returns a reference to the newly constructed object.
2. 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?

If you wrote the code Counter myCounter; without specifying a value, it would have the default value of null. This is because, in C#, reference types (such as classes) are defaulted to null if not explicitly assigned a value. As a result, myCounter would initially point to null before being allocated a reference to a Counter object.

## 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.



MainClass

MyCounter[name,value]

Couter

MyCounter[0] MyCounter[1] MyCounter[2]

MainClass

MyCounter [name,value]

Stack

Heap