3.2P: Answer Sheet

Recall task *2.2P Counter Class* and answer the following questions.

1. How many *Counter* objects were created?

2 counter objects were created

2. Variables declared without the ***new*** keyword are different to the objects created using ***new***. In the ***Main*** function, what is the relationship between the variables initialized with and without the ***new*** keyword?

Variables initialized with the new keyword are stored on the heap, which means they are dynamically allocated and accessed via references. In contrast, variables initialized without the new keyword are typically value types and are stored on the stack. These value types can be accessed directly, without needing a reference.

3. In the ***Main*** function, explain why the statement ***myCounters***[2].***Reset***(); also changes the value of ***myCounters***[0].

because myCounters[2] is equal to myCounters[0].

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

Dynamically allocated memory means that the heap memory size is flexible and can be used for large memory blocks whereas the size of the stack memory is determined at runtime by the compiler and is limited and small compared to heap.
the life of heap memory is managed manually by the programmer but the life of stack memory is managed automatically and is allocated and deallocated as functions are entered and exited.

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

objects are allocated on the heap and local variables are allocated on the stack.

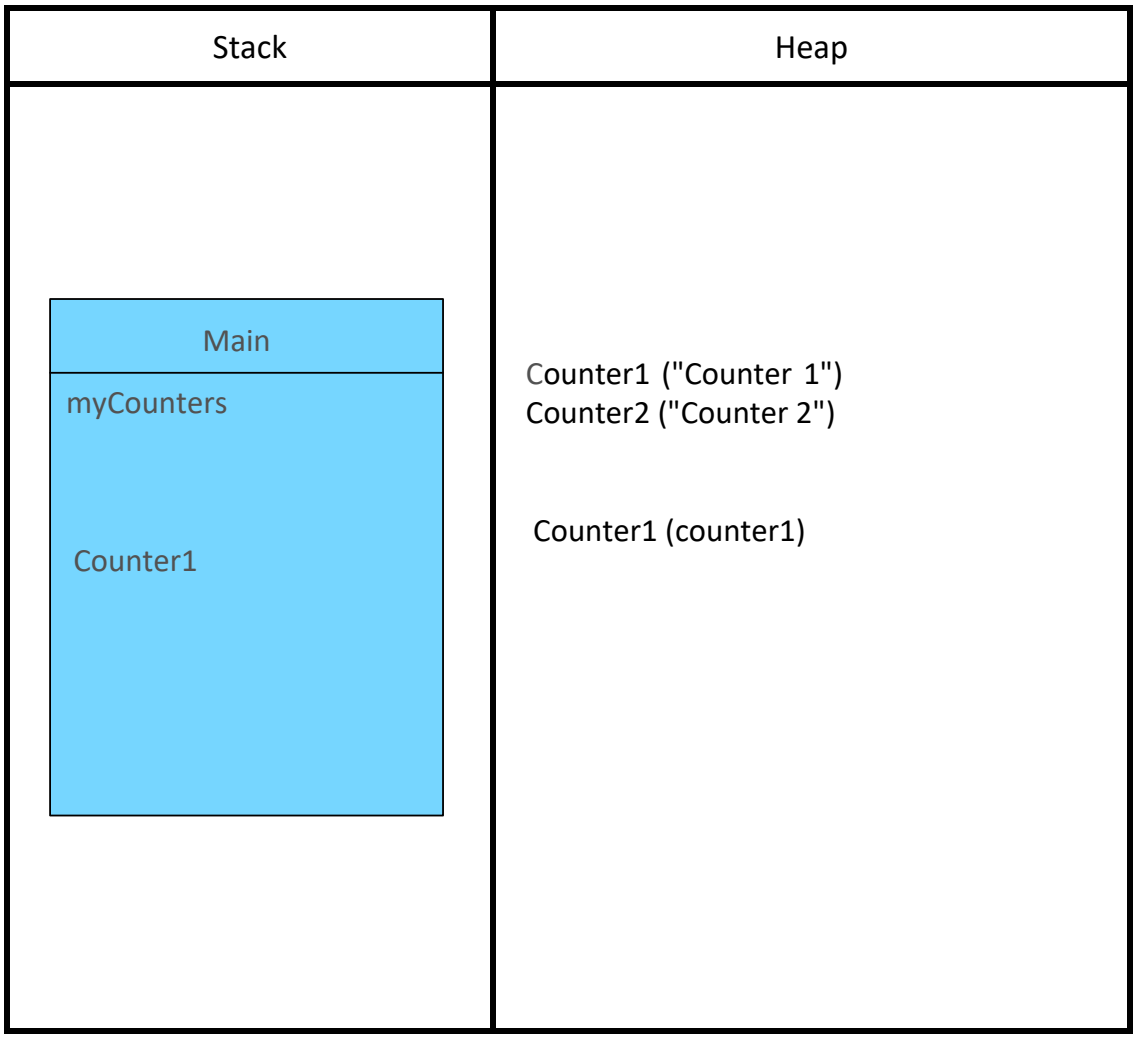
6. What is the meaning of the expression ***new*** *ClassName*(), where *ClassName* refers a class in your application? What is the value of this expression?

The expression new ClassName() creates a new instance (or object) of the class ClassName. This expression calls the class constructor, which initializes the object. The value of this expression is a reference to the newly created object, with its properties set to the default values defined by the constructor.

7. Consider the statement "*Counter* ***myCounter;***". What is the value of ***myCounter*** after this statement? Why?

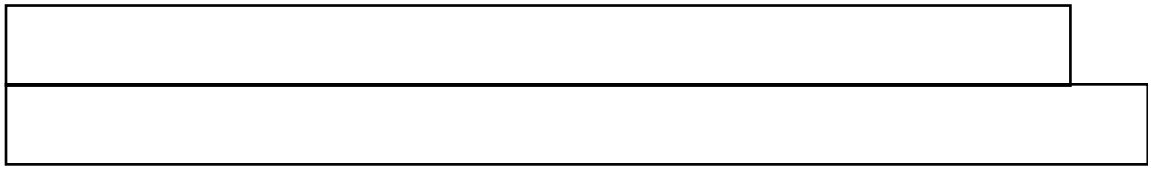
The value of myCounter after the statement Counter myCounter; will be undefined or null, depending on the language, because this statement only declares the variable without instantiating it. No constructor is called, so any initial value (like 0) is not set.

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



9. **If the variable myCounters is assigned to null, then you want to change the value of myCounters[X], where X is the last digit of your student ID, what will happen? Please provide your observation with screenshots and explaination.**

ans . If `myCounters` is set to `null`, attempting to access `myCounters[X]` (where `X` is any valid index) would result in a `NullReferenceException`. This is because `myCounters` no longer points to any array in memory, so dereferencing it causes an error. In C#, accessing a null object reference is not allowed and leads to this



- Null pointer CrowdStrike Bug, https://www.thestack.technology/crowstrike-null-pointer-blamed-rca/
- CrowdStrike Blog, https://www.crowdstrike.com/blog/tech-analysis-channel-file-may-contain-null-bytes/