DotNET C# Assignment

# 1. Data Types:

## a) Briefly describe the two types of dynamic memory that C# and all other languages in the C family of languages support.

The two types of dynamic memory that C# and all other languages in the C family of languages support are stack and heap. The variables and parameters that are local to a function are pushed onto the stack and popped out of the stack when the function call ends and they are no longer needed for computation. In this way, stack memory is automatically allocated and deallocated. On the other hand, heap is a type of dynamic memory must be manually allocated and deallocated. This is accomplished using operators or functions defined by the language.

## b) Which of these types is used to implement function calls in general and recursion in particular?

The run-time stack dynamic memory type is used to implement function calls and recursion. As a function is called, the variables and parameters that are local to it are allocated in the stack. When the function ends, the memory that was allocated in the stack is then deallocated. In a recursive function call the contents of each function call are allocated to the stack until the final function call is reached. Then working backwards from the latest call to the first one, the functions end and the dynamic stack memory is deallocated for the variables and parameters from functions that terminated.

## c) Which of these types is intended to hold linked structures that involve pointers?

The heap dynamic memory type is intended to hold linked structures that involve pointers. In a linked structure, such as a linked list, the size of the structure is dynamic and can change with time, but there is no explicit order to when elements will need to be allocated and deallocated. A stack contains an order with the latest allocated element being the first deallocated element. In a linked list, the fifth element may need to be deallocated and then the first element may need to be deallocated followed by a new element being allocated at the end of the linked list. The order of these elements may also need to be shifted around. This application is better suited to a heap, where each element in the linked list has no relation to the allocation and deallocation of another element. When a new element is added to the linked list, memory is allocated, and when an element is removed from the linked list, memory is deallocated. This principle can be extended to other types of linked structures as well.

## d) Which of these types is intended to hold large objects that are referenced by pointers?

The heap dynamic memory type is intended to hold large objects that are referenced by pointers. In Java, objects and arrays are located on the top level of the heap and are references using pointers. This means that objects and arrays can only be referenced by pointers in Java and cannot by referenced by value.

# 2. Briefly describe the new features of Visual Studio 2017 and C# 7.0.

The release of Visual Studios 2017 brought several new features that developers will find useful. These main features came along with many bug fixes, GUI changes, and quality-of-life updates to visual studios. The major first feature that developers will find helpful is that EditorConfig is now built into Visual Studios 2017 rather than being accessible only through a plugin as it was in Visual Studios 2015. The installer for Visual Studios 2017 also changed, allowing users to only install components that they will use rather than having a larger installation with features that will not be used. A new option also allows users to manage Visual Studios performance and Visual Studios 2017 has live unit testing. The support for JavaScript development has increased and Docker support is now built into Visual Studios 2017. Finally, Visual Studios 2017 allows users to directly install Xamarin for mobile application development. These changes to Visual Studios 2017 reflect the fact that Microsoft is attempting to give Visual Studios a broader scope of use cases. C# 7.0, which was released with Visual Studios 2017, also gained some new features. These features include out variables, tuples and deconstruction, pattern matching, local functions, expanded expression bodied members, ref locals, ref returns, discards, binary literals and digit separators, and throw expressions. These features are aimed at helping developers write cleaner code. For instance, tuples allow developers to store two different data types in one variable and pattern matching allows developers to test expressions for the presence of certain characteristics. Discards can also be useful to clarify the intent of a developer and the ref locals and ref returns give developers more options with regards to implementing references in their code.

# 3. Briefly describe the intended applications of .NET Core and .NET Native.

The intended applications of .NET core are applications that work on multiple platforms or applications that are based on the cloud. This is because .NET core is a modular version of the .NET framework that can run on many different systems and platforms. The intended application of .NET native is developing windows apps. These apps can be downloaded for Microsoft windows devices such as Xbox, PC, Windows smartphones, and other devices. The .NET core platform translates code to an intermediate level before it does just-in-time compilation while .NET native complies directly to native code.

# 4. List all the differences you can think of between a C# class type and a C# struct type.

The differences between a C# class and struct is not initially very clear because they seem to play very similar roles within the language. However, there are several important differences between classes and structs. Firstly, classes are reference types while structs are value types. This then causes classes to be stored in heap memory while structs are be located on stack memory. This is because reference types are stored in heap while value types are stored in stack memory. Another difference is that classes can have constructors and destructors without parameters while structs cannot. This allows classes to be more configurable structures for data abstraction because you can have default constructors without parameters. A difference when using these data structures is that classes use the new keyword to instantiate objects while structs do not need the keyword. Finally, classes can have inheritance while structs cannot. This once again makes classes more optimal for data abstraction. Overall, structs are a simpler version of classes. Structs are sometimes considered purposeless redundancies because they function as more restricted classes, but as discussed, they have several differences that go beyond this idea. Moreover, some may have the notion that C# structs cannot have function members, but this is false. This notion may come from C, where structs exist but methods do not.

# References and Documentation

To understand the new features of Visual Studios 2017 and C# 7.0 as well as the different components of the .NET platform and functionalities of the C# language I used several different sources of documentation. Listed below are the sources that I used to learn about C#, .NET, and the version history of Visual Studios and C#.

1. <https://learn.microsoft.com/en-us/dotnet/csharp/>

- C# documentation

2. <https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history>

- History of C#

3. <https://learn.microsoft.com/en-us/visualstudio/releasenotes/vs2017-relnotes>

- Changes in Visual Studios 2017

4. <https://developer.okta.com/blog/2017/05/03/seven-new-vs-features>

- New features in Visual Studios 2017