**Assignment 01**

**01 Introduction to C# and Data Types**

**Understanding Data Types**

**Test your Knowledge**

**1. What type would you choose for the following “numbers”?**

A person’s telephone number

string

A person’s height

float/double

A person’s age

int

A person’s gender (Male, Female, Prefer Not To Answer)

enum

A person’s salary

decimal

A book’s ISBN

uint

A book’s price

float/double

A book’s shipping weight

float/double

A country’s population

int

The number of stars in the universe

ulong

The number of employees in each of the small or medium businesses in the United Kingdom (up to about 50,000 employees per business)

Int

**2. What are the difference between value type and reference type variables? What is boxing and unboxing?**

difference between value type and reference type variables:

- Value type variables are declared using primitive types, struct, or enum.

- Value type variables are not managed by the garbage collector.

- Value type variables hold the value directly.

- Value type variables are allocated on stack.

- Value type variables cannot be null.

- Reference type variables are declared in a class or interface.

- Reference type variables are cleaned up by the garbage collector.

- A reference type variable hold the memory address to the object.

- Reference type variables are allocated on the heap.

- Reference type variables can accept a null value.

boxing and unboxing:

Boxing refers to converting a reference type variable to value type variable. Unboxing is the opposite process of packaging a

value type variable into a reference type instance. Boxing and unboxing are necessary when converting between the two because

they are stored in different memory locations.

Specifically:

Boxing is the process of converting a value type instance to an object (reference type) instance. This is done by creating

a new object instance on heap and copying the value into it. Boxing is implicit in C#, which means that it happens automatically

when we try to store a value type in an reference type variable.

Unboxing is the process of converting an object back to a value type instance. The value is copied from the object on the heap

to the value type instance on the stack. Unboxing is an explicit operation in C#, meaning that you need to use a cast to perform the conversion.

Unboxing can cause a runtime exception (InvalidCastException) if the object being unboxed is not a boxed value of the value type it is

being unboxed to.

**3. What is meant by the terms managed resource and unmanaged resource in .NET**

managed resources are memory objects handled by the .NET garbage collector. Anything allocated with the new keyword is managed resource.

The GC automatically reclaims the memory occupied by managed resources when they are no longer in use, keeping the application memory-efficient.

On the other hand, unmanaged resources are not directly controlled by the .NET runtime, such as file handles, network sockets, database connections.

The program is responsible for disposal of these resources.

**4. Whats the purpose of Garbage Collector in .NET?**

The garbage collector provides a solution for memory management so that users won't have to deal with potential memory leaks from forgetting to free

an allocated object. .NET Garbage Collector automatically free up memory space that has been allocated to objects no longer needed by the program.

Garbage Collector provides memory safety and frees developers from having to manually release memory. It allows allocated objects on the heap to be managed efficiently,

and recycles memory avaialbe for future use after they're no longer needed.

**Controlling Flow and Converting Types**

**Test your Knowledge**

**1. What happens when you divide an int variable by 0?**

Runtime error. System.DivideByZeroException: 'Attempted to divide by zero.'

**2. What happens when you divide a double variable by 0?**

The program executes, and the result value is inf (double.PositiveInfinity).

**3. What happens when you overflow an int variable, that is, set it to a value beyond its range?**

The overflown number is shown. If the value goes over type limit, it will wrap around to the opposite end of the range.

**4. What is the difference between x = y++; and x = ++y;?**

x = y++ increments y by the end of the statement after addition, ++y increments y at the beginning of the statement before the addition.

**5. What is the difference between break, continue, and return when used inside a loop statement?**

break exits the current level of loop. Continue skips the current iteration and executes the next iteration. Return exits the function.

**6. What are the three parts of a for statement and which of them are required?**

The three parts are for(initializer; condition; iterator){}. Only the condition is required.

**7. What is the difference between the = and == operators?**

= is the assignment operator, which assigns right value to left value.

== evaluates if left and right values are equal and returns a boolean.

**8. Does the following statement compile? for ( ; true; ) ;**

Yes, it's an infinite loop.

**9.What does the underscore \_ represent in a switch expression?**

Underscore represents the discard pattern or wildcard pattern, which handles all remaining cases that are not explicitly defined.

**10. What interface must an object implement to be enumerated over by using the foreach statement ?**

To be enumerated by foreach, an object must implement the IEnumerable inerface, which has a single function IEnumerator GetEnumerator() that need to be defined.

**Assignment 02**

**02 Arrays and Strings**

**Test your Knowledge**

**1. When to use String vs. StringBuilder in C# ?**

String is immutable and StringBuilder is dynamic.

String objects are build from scratch each time a change is made to the string.

StringBuilder objects allocates a buffer space on heap for the string to grow and shrink.

So if there's a lot of add/remove/insertions to be done to the string we should use a StringBuilder.

If the string is static then we can use a simple C# String.

**2. What is the base class for all arrays in C#?**

The base class for all arrays in C# is System.Array, an abstract class.

**3.How do you sort an array in C#?**

Array.Sort(nameOfTheArray).

**4. What property of an array object can be used to get the total number of elements in an array?**

nameofTheArray.length.

**5. Can you store multiple data types in System.Array?**

No. Elements must be the same type in a System.Array.

However it's possible to make a generic object[] to approximate the effect of storing different data types.

**6. What’s the difference between the System.Array.CopyTo() and System.Array.Clone()?**

System.Array.CopyTo() copies an array into an existing array.

System.Array.Clone() creates a new array object cotaining all the values in the original one.

Additionally, CopyTo() creates a shallow copy of the elements. Clone() preserves the reference types of the original array.

In CopyTo(), we can a position to start copying from. Clone() copies the entire array.