Question 1.1

Welcome to Learning C# with Unity

First, let us start with what exactly C# is, C# is a high-level language that is statically and strongly typed. C# was created by Microsoft in 2000 and is part of the C family of coding.

Now we understand the type of language that C# is we can continue to build on this knowledge. First, we are going to output a string to the console, however, if you run the code below, you will encounter errors, this is due to no semicolon being at the end of the statement meaning the compiler does not know this is the end of the statement. Building upon this, the correct spelling of functions and variables is paramount else the compiler does not know what you are referencing. There are two of these errors in the debug.log function, if you succeed in correcting them you should see ‘Hello World’ outputted on the right of your screen.

1. debug.log(“Hello World”)

Question 1.2

Now we can output to the console, we can start building our knowledge. First, we are going to add to numbers together, this is done like real life, but the main difference is a variable is defined with an integer value before numbers can be added together. In C#, it would look like this.

1. int a = 5, b = 5, c;
2. c = a + b;

The code above would give us a result of 10, Try out the above code and see if you can find other ways of adding numbers together.

Question 1.3

Subtraction of integers works nearly the same as the addition of integers. In this lesson you will see a new technique of subtracting numbers which may also be used to add numbers.

Instead of subtracting two integers from one another, we could subtract by doing the following:

1. int a = 5;

2. a -= 5;

This code is saying that the new value of 'a' is equal to the current value of 'a' minus 5.

Try out the code and output the result to the console.

Question 1.4

Multiplication of integers works just like the real world; in C# we use the asterisk (\*) to represent a multiplication. Additionally, multiplication within C# has to follow the same PEMDAS system as the real world. For example, the following code would output a result of 30, this is because the multiplication is being done first and the compiler is calculating 5 + 25.

1. int a = 5 + 5 \* 5;

However, if we add a pair of parentheses around 5 + 5 to make the code look like this:

1. int a = (5 + 5) \* 5;

We would now get a result of 50 due to the compiler performing the operation in the parenthesis first.

Now we understand multiplication, try placing a pair of parentheses in this code so it no longer outputs 34 but instead outputs 64.

PEMDAS is an acronym for Parentheses, Exponential, Multiplication, Division, Addition and Subtraction. In the United Kingdom this system may be know as BODMAS (Brackets, Orders, Division, Multiplication, Addition, Subtraction). Both systems can be used as the work the same and is to be used when there are two or more operations in a single expression.

Question 1.5

The division of integers follows the same rules as the multiplication, in C# we represent a division with the forward slash (/). Division still follows the order of operation formally mentioned in the previous question. As there is not much more to say on division why not try solving the following equation so it so it no longer outputs -4 but now -1

1. int a = -2 \* 1 \* 4 - 2 / 2 + 6 + 2 - 3;