**Understanding Data Types**

Question 1:

A person’s telephone number: uint

A person’s height: byte

A person’s age: byte

A person’s gender: byte

A person’s salary: uint

A book’s ISBN: ulong

A book’s price: ushort

A book’s shipping weight: byte

A country’s population: uint

The number of stars in the universe: ulong

The number of employees in each of the small or medium businesses in the UK: ushort

Question 2:

Value types directly hold the value, they cannot accept null value, use struct or enum to create, not collected by garbage collector and stored in stack memory.

Reference types hold the memory address for the values. They use class, interface or array to create, collected by garbage collector and stored in heap memory.

Boxing and unboxing describe the ability to convert a value type to a reference type and vice versa. They are expensive for computation cost.

Question 3:

Managed refers to anything within the .NET sandbox. This includes all .NET Framework classes.

Unmanaged refers to the wilderness outside the .NET sandbox. This includes anything that is returned to you through calls to Win32 API functions.

Question 4:

Garbage collector is used for automatic memory management without need to manually release memory. It also allocates objects on managed heap efficiently.

**Controlling Flow and Converting Types**

Question 1:

A DivideByZeroException is thrown.

Question 2:

If the double variable is positive, a positive infinity is returned. If the double variable is zero, NaN is returned. If the double variable is negative, a negative infinity is returned.

Question 3:

The Visual Studio will stop me from setting a constant value beyond its range but if I overflow an inbound int variable by adding a positive number, C# won’t stop me by throwing any exception but overflow the int by incrementing from the lowest value of int.

Question 4:

x = ++y means that when the code is executing it will first do y = y + 1 and then set x = y.

x = y++ means that when then code is executing it will first read it, set x = y and do the y = y + 1 after y has been read.

Question 5:

The break statement results in the termination of the loop, it will come out of the loop and stops further iterations. The continue statement stops the current execution of the iteration and proceeds to the next iteration. The return statement takes you out of the method. It stops executing the method and returns from the method execution.

Question 6:

The three parts are initializer, condition and iterator. All the three parts are optional.

Question 7:

The “=” is an assignment operator which is used to assign the value on the right to the variable on the left while the “==” operator checks whether the two given operands are equal or not.

Question 8：

It compiles but it is an infinite for loop.

Question 9:

The underscore “\_” character replaces the default keyword to signify that it should match anything if reached.

Question 10:

The IEnumerable interface permits enumeration by using a foreach loop.

**Practice loops and operators**

Question 1:

The for loop becomes a infinite loop because of byte overflow.

I can add checked {} outside the for loop to throw an exception when overflow happened.