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

A person’s telephone number

string

A person’s height

float or 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

string

A book’s price

decimal

A book’s shipping weight

float or double

A country’s population

long

The number of stars in the universe

BigInteger

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?

**Value Type:**

* **Definition:** Value types directly contain their data. Each variable of a value type has its own copy of the data.
* **Storage:** Stored in the stack.
* **Example:** int, float, double, struct, enum
* **Behavior:** When you assign a value type to another value type, a copy of the value is created.

**Reference Type:**

* **Definition:** Reference types store references to their data (objects). The variables of a reference type can refer to the same object.
* **Storage:** Stored in the heap.
* **Example:** class, interface, delegate, string, array
* **Behavior:** When you assign a reference type to another reference type, both variables refer to the same object. Modifying one variable will affect the object referenced by the other variable.

**Boxing and Unboxing:**

* **Boxing:** The process of converting a value type to a reference type. It involves wrapping the value type inside an object or any interface type implemented by this value type.
* **Unboxing:** The process of converting a reference type back to a value type. It involves extracting the value type from the object.

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

**Managed Resource:**

* **Definition:** Resources that are managed by the .NET runtime's garbage collector. These include memory, threads, and handles managed by the CLR.
* **Examples:** Instances of classes, arrays, and strings.

**Unmanaged Resource:**

* **Definition:** Resources that are not managed by the .NET runtime and need explicit cleanup. These include resources that the .NET garbage collector does not handle directly.
* **Examples:** File handles, database connections, COM objects, and memory allocated using native code.

4. What is the purpose of Garbage Collector in .NET?

**Garbage Collector (GC):**

* **Purpose:** The GC is responsible for automatic memory management in .NET. It helps to ensure efficient use of memory by reclaiming the memory occupied by objects that are no longer in use.
* **Functions:**
  + **Automatic Memory Management:** Frees developers from manual memory management tasks, reducing the risk of memory leaks and fragmentation.
  + **Memory Allocation:** Allocates memory for new objects efficiently.
  + **Garbage Collection:** Periodically identifies and frees up memory occupied by objects that are no longer accessible by the application.
  + **Finalization:** Provides a mechanism for cleanup of unmanaged resources by allowing the implementation of a finalizer (destructor) that runs before an object is reclaimed.

By managing memory efficiently and automatically, the GC helps in maintaining application performance and stability.

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

When you divide an int variable by 0, a DivideByZeroException is thrown at runtime. This is because dividing an integer by zero is undefined and the .NET runtime enforces this by throwing an exception.

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

When you divide a double variable by 0, the result is either positive infinity (Infinity), negative infinity (-Infinity), or NaN (Not a Number), depending on the sign of the numerator:

* double.PositiveInfinity if the numerator is positive.
* double.NegativeInfinity if the numerator is negative.
* NaN if the numerator is zero.

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

When an int variable overflows (exceeds its range), the behavior depends on whether overflow checking is enabled:

* **Unchecked Context (default):** The value wraps around to the minimum value and continues from there (modulus arithmetic). For example, int.MaxValue + 1 becomes int.MinValue.
* **Checked Context:** A System.OverflowException is thrown if the operation results in an overflow.

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

x = y++;:

This is a post-increment operation. y is assigned to x first, and then y is incremented by 1.

x = ++y;: This is a pre-increment operation. y is incremented by 1 first, and then the new value of y is assigned to x.

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

**break:** Terminates the loop immediately and transfers control to the statement following the loop.

**continue:** Skips the rest of the current iteration and proceeds with the next iteration of the loop.

**return:** Exits the method in which it is called, regardless of whether it is inside a loop.

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

The three parts of a for statement are:

1. **Initialization:** Executed once at the beginning and is used to initialize the loop control variable.
2. **Condition:** Evaluated before each iteration, and the loop continues as long as the condition is true.
3. **Iteration:** Executed at the end of each iteration and is typically used to update the loop control variable.

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

**= (Assignment Operator):** Assigns the value on its right to the variable on its left.

**== (Equality Operator)** Compares the values on its left and right sides and returns true if they are equal, otherwise false`.

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

Yes, the statement compiles. It creates an infinite loop because the condition true is always satisfied, and there are no initialization or iteration parts.

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

In a switch expression, the underscore \_ is used as a discard pattern and represents the default case that matches any value not matched by other patterns. It is similar to the default case in a traditional switch statement.

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

An object must implement the IEnumerable or IEnumerable<T> interface to be enumerated over by using the foreach statement.