Question 1

[**Managed**] code, as seen in languages like C# and F#, runs within an environment, such as the Common Language Runtime (CLR), that provides benefits like memory management and security. [**Native**] code, written in languages like C and C++, compiles directly to [**machine**] code, offering low-level access to hardware and faster performance.

Question 2

Given the code snippet:

```
int[] evenNumbers = new int[5];
for (int i = 0; i < 5; i++)
{
   evenNumbers[i] = // [complete assignment instruction]
}
```

Complete the instruction within the loop so that the even numbers from 2 to 10 are assigned to the elements of the array.

Example of a correct response

When:

```
evenNumbers[i] = 2 * (i + 1);
```

Is used inside the for loop, even numbers are assigned to the evenNumbers array, ranging from 2 to 10.

Question 3

Explain the difference between *object* and *class* in approximately 100 words.

Example of a correct response

In object-oriented programming, a class is a blueprint or template that defines the structure and behaviour of objects. It encapsulates attributes (data) and methods (behaviours) that are common to a group of objects. Objects, on the other hand, are instances created from a class. They represent real-world entities and can have unique data values while inheriting the behaviour defined in the class. In essence, a class defines the characteristics and actions that objects of that class will possess, allowing for code reusability and organisation, making it a fundamental concept in building maintainable software systems.

Question 4

Consider the *MathOperations* class with the provided methods:

```
public class MathOperations
 {
    public int Add(int a, int b)
    {
       return a + b;
    }

    public double Add(double a, double b)
    {
       return a + b;
    }
 }
```

Explain the purpose of having multiple methods that perform similar tasks and how this design benefits code organisation and flexibility (in approximately 100 words). Provide an example of how you would use these methods with different arguments types and their expected outcomes.

Example of a correct response

The *MathOperations* class employs method overloading by providing multiple methods for addition—one for integers and another for doubles. This approach enhances code organisation and flexibility by allowing developers to

define different implementations for distinct data types with the same method name. Method overloading simplifies an object interface, promotes code reusability, and improves code readability.

For example, we can use the *Add* methods to perform addition with different data types, ensuring the appropriate method is called based on the argument types:

```
MathOperations math = new MathOperations();
int result1 = math.Add(3, 4);       // Calls the int Add method
double result2 = math.Add(2.5, 3.7); // Calls the double Add method
```

Question 5

Explain the roles of the stack and heap memory during the execution of a program. What are the key differences between these memory regions, and in what scenarios are they used? Use approximately 100 words.

Example of a correct response

In the context of C# and many other programming languages, memory management is a fundamental aspect of efficient program execution. It revolves around two primary memory regions: the stack and the heap.

The stack is primarily responsible for managing method calls and local variables. It operates on a last-in, first-out (LIFO) basis, meaning that local variables are immediately deallocated when a method exits. This makes it well-suited for efficiently handling simple data types, such as integers and references. Stack memory allocation is deterministic and straightforward, ensuring efficient memory cleanup.

On the other hand, the managed heap serves as a dynamic memory region for managing objects and data with longer lifetimes. It accommodates dynamic memory allocation and deallocation, making it suitable for objects, arrays, and data structures with unpredictable durations. While the heap offers greater flexibility, it also involves more time and overhead in memory allocation compared to the stack, as it managed by the garbage collector.

Question 6

```
public class Program
{
   public static void Main()
   {
      bool isValidInput = false;
      while (!isValidInput)
      {
         Console.Write("Enter a double: ");
         string userInput = Console.ReadLine();
         try
         {
            double number = Convert.ToDouble(userInput);
            Console.WriteLine("You entered: " + number);
            isValidInput = true;
         }
         catch (FormatException ex)
         {
            Console.WriteLine("Format Exception: " + ex.Message + " Please try again.");
         }
         catch (OverflowException ex)
         {
            Console.WriteLine("Overflow Exception: " + ex.Message + " Please try again.");
         }
         catch (Exception)
         {
            Console.WriteLine("An unexpected error occurred. Please try again.");
         }

         Console.WriteLine("Exiting the Program");
      }
   }
}
```

In the code snippet provided, which type of exception is handled by the

```
catch (FormatException ex)
```

Correct response

Invalid input format (e.g., entering a non-numeric value).

The catch would be executed when the string userInput contains a value that does not represent a number and hence cannot be converted to a double by the call to Convert.ToDouble(userInput).

Question 7

```
class Program
{
   public static void Main(string[] args)
   {
      int x = 5;
      int y = 5;
      if (x == y)
         Console.WriteLine("x is equal to y");
      Product product1 = new Product("ABC123", "Widget", 10.0);
      Product product2 = new Product("ABC123", "Widget", 10.0);
      if (product1 == product2)
         Console.WriteLine("product1 is equal to product2");
   }
}
```

Given the above code, what will be the output of the execution, and why?

Correct response

"x is equal to y" will be printed, but "product1 is equal to product2" will not be printed because the comparison of product1 and product2 is done on references.

Further Questions

The In-class test is designed to match the type and style of questions you might be asked during an actual programming job interview. In addition to the mock-up ICT, further examples of questions that can help you prepare for the assessment may also be found online.