C#AND.NET PROGRAMMING
MATERIAL
2 MARKS

1. C# data types are mainly divided into:
   1. **Value Types** – Examples: int, float, double, char, bool.
   2. **Reference Types** – Examples: string, object, array, class.

These define how data is stored and accessed in memory.

2. A **class** in C# is a blueprint for creating objects. It defines properties and methods.
An **object** is an instance of a class that holds actual values.

```
class Car {
    public string color;
}


Car myCar = new Car();
myCar.color = "Red";
```

3. type conversion in C# is changing one data type to another. It is of two types:
   Implicit Conversion – Done automatically (e.g., int to float).
   Explicit Conversion (Casting) – Done manually using cast operator.

   Example:
   ```
   int num = 10;
   float f = num;          // Implicit
   double d = (double)num;  // Explicit
   ```

4. Loops in C# are used to execute a block of code repeatedly. Common loops are:

1. **for loop**
2. **while loop**
3. **do-while loop**
4. **foreach loop**

Example:

```
for (int i = 1; i <= 5; i++) {
   Console.WriteLine(i);
}
```

5. An **array** in C# is a collection of elements of the same type stored in a contiguous memory location. It allows storing multiple values in a single variable.
   Example:
   ```
   int[] numbers = { 10, 20, 30, 40 };
   Console.WriteLine(numbers[0]); // Output: 10
   ```

6. A **one-dimensional array** in C# is a linear collection of elements of the same type, accessed using a single index.
   Example:
   ```
   int[] marks = new int[3] { 85, 90, 95 };
   Console.WriteLine(marks[1]); // Output: 90
   ```

7. **Polymorphism** in C# means the ability of an object to take many forms. It allows methods to behave differently based on the object that calls them.
   Types:
   Compile-time (Method Overloading)
   Run-time (Method Overriding)
   Example:
   ```
   class Animal {
      public virtual void Sound() {
         Console.WriteLine("Animal sound");
      }
   }
   ```

```
class Dog : Animal {
    public override void Sound() {
        Console.WriteLine("Bark");
    }
}
```

8. **Overloading** in C# means having multiple methods with the same name but different parameters within the same class. It is a type of **compile-time polymorphism**.

   Example:
   ```
   class Math {
       public int Add(int a, int b) {
           return a + b;
       }

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

9. An **interface** in C# is a contract that defines method signatures without implementation. A class that implements an interface must define all its methods.

   Example:
   ```
   interface IAnimal {
       void Speak();
   }

   class Dog : IAnimal {
       public void Speak() {
           Console.WriteLine("Bark");
       }
   }
   ```

10. **Operator Overloading** in C# allows you to redefine the meaning of an operator (like +, -) for user-defined types (e.g., classes).

    Example
    ```
    class Complex {
        public int real, imag;
    ```

```
    public static Complex operator +(Complex c1, Complex c2) {
        return new Complex { real = c1.real + c2.real, imag = c1.imag +
c2.imag };
    }
}
```

11. **Machine language** is the lowest-level programming language that consists of binary code (0s and 1s). It is directly understood by a computer's CPU and does not require translation.
It is specific to the architecture of the processor.
Example:
For a CPU, machine language instructions might look like 10101000 to perform a specific operation.

12. **.NET programming** refers to software development using the **.NET Framework** or **.NET Core**, a platform developed by Microsoft. It supports multiple programming languages like C#, VB.NET, and F#. .NET provides a comprehensive set of libraries and tools for building applications, including web, desktop, mobile, and cloud-based applications.
Example:
A simple C# program in .NET:

```
using System;
class Program {
    static void Main() {
        Console.WriteLine("Hello, .NET!");
    }
}
```

13. **Arithmetic operations** in C# are basic mathematical operations that can be performed on numeric data types. The common arithmetic operators are:

1. **Addition (+)** – Adds two numbers.
2. **Subtraction (-)** – Subtracts one number from another.
3. **Multiplication (*)** – Multiplies two numbers.
4. **Division (/)** – Divides one number by another.
5. **Modulus (%)** – Returns the remainder of a division.

14. The switch case statement in C# is used to execute one out of multiple possible blocks of code based on the value of an expression. It simplifies complex if-else conditions.
Example:

```
int day = 2;
switch (day) {
    case 1:
        Console.WriteLine("Monday");
        break;
    case 2:
        Console.WriteLine("Tuesday");
        break;
    case 3:
        Console.WriteLine("Wednesday");
        break;
    default:
        Console.WriteLine("Invalid day");
        break;
}
```

15. In C#, types are classified into two categories:

1. **Value Types**: Store data directly in memory. Examples include int, float, bool, char. They are stored in the stack.
2. **Reference Types**: Store references (addresses) to data in memory, rather than the data itself. Examples include string, class, array, object. They are stored in the heap.

Example:

```
int a = 10;  // Value type
string str = "Hello";  // Reference type
```