Uttaranchal Institute of Technology

Mid Term Examination – Odd Semester (2025–26)

Programme: B.Tech (CSE)

Course: Visual Programming and .NET Technologies

Course Code: TCS-302

SECTION – A

(Very Short Answer Type Questions – 1 Mark Each)

(Answers are concise and exam-oriented)

Q1(a) Define the role of the .NET runtime environment.

Answer:

The .NET runtime environment (CLR) manages code execution, memory management, garbage collection, security, and exception handling in .NET applications.

Q1(b) List two different types of assemblies in the .NET Framework.

Answer:

1. Private Assembly
2. Shared Assembly

Q1(c) Write the basic syntax of a while loop in C#.

Answer:

```
while(condition)
{
    // statements
}
```

Q1(d) Explain the use of the static keyword in a method definition.

Answer:

The `static` keyword allows a method to be accessed without creating an object of the class.

Q1(e) State the purpose of the get accessor in C# properties.

Answer:

The `get` accessor is used to read or retrieve the value of a property.

Q1(f) Name two types of inheritance supported by C#.

Answer:

1. Single Inheritance
2. Multilevel Inheritance

☑ SECTION – B

(Short Answer Type Questions – 4 Marks Each)

(Detailed explanation + Code + Output)

Q2(a) Illustrate how the architecture of the .NET Framework supports application development across multiple languages.

Answer:

Definition:

The .NET Framework supports **multiple programming languages** through a common architecture.

Architecture Components:

1. **CLR (Common Language Runtime)** – Executes code
2. **CTS (Common Type System)** – Ensures type compatibility
3. **CLS (Common Language Specification)** – Enables language interoperability
4. **IL (Intermediate Language)** – Common compiled code

Example Code (C#):

```
int a = 10;
Console.WriteLine(a);
```

Explanation:

This code can interact with components written in VB.NET or F# because all compile to **IL**.

Output:

```
10
```

OR

Q2(a) How do you resolve name clashes using fully qualified names?

Answer:

Explanation:

Name clashes occur when two classes have the same name in different namespaces.

Fully qualified names specify the complete namespace path.

Example Code:

```
System.Console.WriteLine("Hello");
```

Output:

```
Hello
```

Q2(b) Demonstrate the use of nested if-else statements in C# with a program that checks grade categories.

Answer:

```
using System;

class Program
{
    static void Main()
    {
        int marks = 85;

        if (marks >= 90)
            Console.WriteLine("Grade A");
        else if (marks >= 75)
            Console.WriteLine("Grade B");
        else if (marks >= 60)
            Console.WriteLine("Grade C");
        else
            Console.WriteLine("Fail");
    }
}
```

Output:

```
Grade B
```

OR

Q2(b) Differentiate between default and parameterized constructors in C# with sample code.

Default Constructor	Parameterized Constructor
No parameters	Takes parameters
Initializes default values	Initializes custom values

Example Code:

```
class Student
{
    int id;

    public Student()  // Default
    {
        id = 0;
    }

    public Student(int x) // Parameterized
    {
        id = x;
    }
}
```

Q2(c) Explain how compile-time and run-time polymorphism are achieved in C#.

Answer:

Compile-Time Polymorphism (Method Overloading)

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

Run-Time Polymorphism (Method Overriding)

```
class A
{
    public virtual void Show() { Console.WriteLine("A"); }
}

class B : A
{
    public override void Show() { Console.WriteLine("B"); }
}
```

Output:

```
B
```

OR

Q2(c) Implement a generic class in C# that stores a list of elements and returns the count.

```
using System;
using System.Collections.Generic;

class MyList<T>
{
    List<T> items = new List<T>();

    public void Add(T item)
    {
        items.Add(item);
    }

    public int Count()
    {
        return items.Count;
    }
}

class Program
{
    static void Main()
    {
        MyList<int> list = new MyList<int>();
        list.Add(10);
        list.Add(20);

        Console.WriteLine(list.Count());
    }
}
```

Output:

2

☑ SECTION – C

(Long Answer Type Questions – 6 Marks Each)

(Very detailed + Code + Output)

Q3(a) Develop a simple DLL in .NET and demonstrate how to install and reference it in an application.

Answer:

Step 1: Create DLL

```
namespace MyDLL
{
    public class Message
    {
        public string SayHello()
        {
            return "Hello from DLL";
        }
    }
}
```

Compile as **Class Library**.

Step 2: Use DLL

```
using MyDLL;

class Program
{
    static void Main()
    {
        Message m = new Message();
        Console.WriteLine(m.SayHello());
    }
}
```

Output:

```
Hello from DLL
```

OR

Q3(a) Demonstrate how nested namespaces are created and accessed in .NET.

```
namespace College
{
    namespace CS
    {
        class Student
        {
            public void Show()
            {
                Console.WriteLine("CS Student");
            }
        }
```

```
    }
}

class Program
{
    static void Main()
    {
        College.CS.Student s = new College.CS.Student();
        s.Show();
    }
}
```

Output:

```
CS Student
```

Q3(b) Construct a C# program that defines a structure for "Book" and stores details of multiple books using an array.

```
using System;

struct Book
{
    public string title;
    public int price;
}

class Program
{
    static void Main()
    {
        Book[] b = new Book[2];
```

```
        b[0].title = "C#";
        b[0].price = 500;

        b[1].title = "ASP.NET";
        b[1].price = 600;

        for (int i = 0; i < 2; i++)
        {
            Console.WriteLine(b[i].title + " " + b[i].price);
        }
    }
}
```

Output:

```
C# 500
ASP.NET 600
```

OR

Q3(b) Explain the concept of boxing and unboxing in C# with practical examples.

Definition:

- **Boxing:** Value type → Object
- **Unboxing:** Object → Value type

Example Code:

```
int a = 10;
object obj = a;       // Boxing
int b = (int)obj;     // Unboxing
```

```
Console.WriteLine(b);
```

Output:

```
10
```