DEPARTMENT OF CSE(HONORS)
SUBJECT CODE: 22CS2240F-
.NET PROGRAMMMING(EPAM)

Lab 1: Tasks on C# Basics Concepts

Date of the Session: //______________________________Time of the Session:______to______

Learning outcomes:

- Familiarity with C# Basic concepts.
- Outcome related to second session

PRE-LAB

1. What are the arithmetic Operators and Conditional statements in C#

Solution:
Arithmetic Operators in C#:
Addition: Adds two operands.
Example: x + y

Subtraction : Subtracts the second operand from the first.
Example: x - y

Multiplication: Multiplies two operands.
Example: x * y

Division: Divides the first operand by the second.
Example: x / y

Modulus: Returns the remainder of a division operation.
Example: x % y

Increment : Increases the value of an operand by 1.
Example: x++ or ++x

Decrement: Decreases the value of an operand by 1.
Example: x-- or --x

Conditional Statements in C#:
if Statement
Executes a block of code if a condition evaluates to true.

if-else Statement
Provides an alternative execution path if the condition is false.
switch Statement
Tests a single variable against multiple cases. It is an alternative to long if-else if chains for matching discrete values.

2. Answer the following

(i) What is Boxing and Un-Boxing with Example.

Solution:

Boxing is the process of converting a value type (e.g., int, double) into a reference type (object). It happens implicitly when assigning a value type to an object variable. For example:

```
int num = 10;  // Value type
object obj = num;  // Boxing
```

Unboxing is the reverse process, where a boxed object is explicitly converted back into a value type. It requires type casting. For example:

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

IN-LAB:

1. Write a C# code to implement the simple calculator?

TASK1: It's required to create a simple calculator with addition and subtraction operations for two integer numbers

For example, how to find the sum of given integer values **a** and **b**. You have a skeleton code:

```
 public static int Add(int a, int b)
{
      //TODO Delete line below and write your own solution
       throw new NotImplementedException();
}
```

Solution:

```
using System;

class SimpleCalculator
{
   // Method to add two integers
   public static int Add(int a, int b)
   {
      return a + b;
   }

   // Method to subtract two integers
   public static int Subtract(int a, int b)
   {
      return a - b;
   }

   static void Main()
   {
      Console.WriteLine("Simple Calculator");

      // Input two integers
      Console.Write("Enter the first number: ");
      int num1 = Convert.ToInt32(Console.ReadLine());

      Console.Write("Enter the second number: ");
      int num2 = Convert.ToInt32(Console.ReadLine());

      // Perform addition
      int sum = Add(num1, num2);
      Console.WriteLine($"Sum: {num1} + {num2} = {sum}");

      // Perform subtraction
      int difference = Subtract(num1, num2);
      Console.WriteLine($"Difference: {num1} - {num2} = {difference}");
   }
}
```

Expected output:

```
Simple Calculator
Enter the first number: 10
Enter the second number: 5
Sum: 10 + 5 = 15
Difference: 10 - 5 = 5
```

2. Write a C# code to solve the TASK2 and TASK3.

TASK2: For a given integer *n* calculate the value which is equal to:

1. squared number, if its value is strictly positive;
2. modulus of a number, if its value is strictly negative;
3. zero, if the integer n is zero.

Example

n=4 result=16

n=-5 result=5

n=0 result=0

TASK3: Find the maximum integer, that can be obtained by numbers of an arbitrary three-digit positive integer *n* permutation (100<=n<=999).

Example

n=165 result=651

Solution:

```
using System;

class Task2Calculator
{
    public static int CalculateValue(int n)
    {
        if (n > 0)
        {
            return n * n; // Squared number if n is positive
        }
        else if (n < 0)
        {
            return Math.Abs(n); // Modulus of the number if n is negative
        }
        else
        {
            return 0; // Return zero if n is zero
        }
    }

    static void Main()
    {
        // Input integer n
        Console.Write("Enter an integer n: ");
        int n = Convert.ToInt32(Console.ReadLine());

        // Calculate the result based on the condition
        int result = CalculateValue(n);
        Console.WriteLine($"Result: {result}");
    }
}
```

Expected output:

```
Enter an integer n: 4
Result: 16

Enter an integer n: -5
Result: 5

Enter an integer n: 0
Result: 0
```

TASK :3

```
using System;
using System.Linq;

class Task3Calculator
{
    public static int GetMaxPermutedValue(int n)
    {
        // Convert integer to string to easily manipulate digits
        string str = n.ToString();

        // Sort digits in descending order to form the largest possible number
        var sortedDigits = str.OrderByDescending(c => c).ToArray();

        // Convert sorted digits back to integer
        int maxNumber = int.Parse(new string(sortedDigits));
        return maxNumber;
    }

    static void Main()
    {
        // Input integer n (a three-digit number)
        Console.Write("Enter a three-digit integer n: ");
        int n = Convert.ToInt32(Console.ReadLine());

        // Ensure that the input is a valid three-digit number
        if (n >= 100 && n <= 999)
        {
            // Get the maximum value from permutations
            int result = GetMaxPermutedValue(n);
            Console.WriteLine($"Maximum number from permutations: {result}");
        }
        else
        {
            Console.WriteLine("Please enter a valid three-digit number.");
        }
    }
}
```

EXPECTED OUTPUT:

```
Enter a three-digit integer n: 165
Maximum number from permutations: 651
```

POST-LAB

1. Implement a proper calculator with all the functionalities like addition, subtraction, multiplication, division and square root.

Solution:

```
using System;

class Calculator
{
    // Method for addition
    public static double Add(double a, double b)
    {
        return a + b;
    }

    // Method for subtraction
    public static double Subtract(double a, double b)
    {
        return a - b;
    }

    // Method for multiplication
    public static double Multiply(double a, double b)
    {
        return a * b;
    }

    // Method for division
    public static double Divide(double a, double b)
    {
        if (b == 0)
        {
            Console.WriteLine("Error: Division by zero is not allowed.");
            return double.NaN; // Return NaN (Not a Number) to indicate error
        }
        return a / b;
    }

    // Method for square root
    public static double SquareRoot(double a)
    {
        if (a < 0)
        {
            Console.WriteLine("Error: Cannot take the square root of a negative number.");
            return double.NaN; // Return NaN if input is negative
        }
        return Math.Sqrt(a);
    }

    static void Main()
    {
        Console.WriteLine("Simple Calculator with Addition, Subtraction, Multiplication,
Division, and Square Root");

        while (true)
        {
            Console.WriteLine("\nSelect an operation:");
            Console.WriteLine("1. Add");
            Console.WriteLine("2. Subtract");
            Console.WriteLine("3. Multiply");
            Console.WriteLine("4. Divide");
            Console.WriteLine("5. Square Root");
            Console.WriteLine("6. Exit");
```

```
int choice = Convert.ToInt32(Console.ReadLine());

if (choice == 6)
{
    Console.WriteLine("Exiting the calculator.");
    break;
}

double num1, num2, result;

switch (choice)
{
    case 1: // Addition
        Console.Write("Enter first number: ");
        num1 = Convert.ToDouble(Console.ReadLine());
        Console.Write("Enter second number: ");
        num2 = Convert.ToDouble(Console.ReadLine());
        result = Add(num1, num2);
        Console.WriteLine($"Result: {result}");
        break;

    case 2: // Subtraction
        Console.Write("Enter first number: ");
        num1 = Convert.ToDouble(Console.ReadLine());
        Console.Write("Enter second number: ");
        num2 = Convert.ToDouble(Console.ReadLine());
        result = Subtract(num1, num2);
        Console.WriteLine($"Result: {result}");
        break;

    case 3: // Multiplication
        Console.Write("Enter first number: ");
    num1 = Convert.ToDouble(Console.ReadLine());
    Console.Write("Enter second number: ");
    num2 = Convert.ToDouble(Console.ReadLine());
    result = Multiply(num1, num2);
    Console.WriteLine($"Result: {result}");
    break;

case 4: // Division
    Console.Write("Enter first number: ");
    num1 = Convert.ToDouble(Console.ReadLine());
    Console.Write("Enter second number: ");
    num2 = Convert.ToDouble(Console.ReadLine());
    result = Divide(num1, num2);
    if (!double.IsNaN(result))  // Only display if division was successful
    {
        Console.WriteLine($"Result: {result}");
    }
    break;

case 5: // Square Root
    Console.Write("Enter a number: ");
    num1 = Convert.ToDouble(Console.ReadLine());
    result = SquareRoot(num1);
    if (!double.IsNaN(result))  // Only display if valid square root
    {
        Console.WriteLine($"Square Root: {result}");
    }
    break;

default:
    Console.WriteLine("Invalid selection. Please choose a valid operation.");
    break;
```

```
            }
          }
        }
      }
```

EXPECTED OUTPUT:

```
Simple Calculator with Addition, Subtraction, Multiplication, Division, and Square Root

Select an operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Square Root
6. Exit
1
Enter first number: 10
Enter second number: 5
Result: 15

Select an operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Square Root
6. Exit
4
Enter first number: 10
Enter second number: 0
Error: Division by zero is not allowed.
Result: NaN

Select an operation:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Square Root
6. Exit
5
Enter a number: 16
Square Root: 4
```

(For Evaluators use only)

Comment of the Evaluator(if Any)	Evaluator's Observation Marks Secured:________out of________ Full Name of the Evaluator: Signature of the Evaluator: Date of Evaluation