LAB-2

2100032064

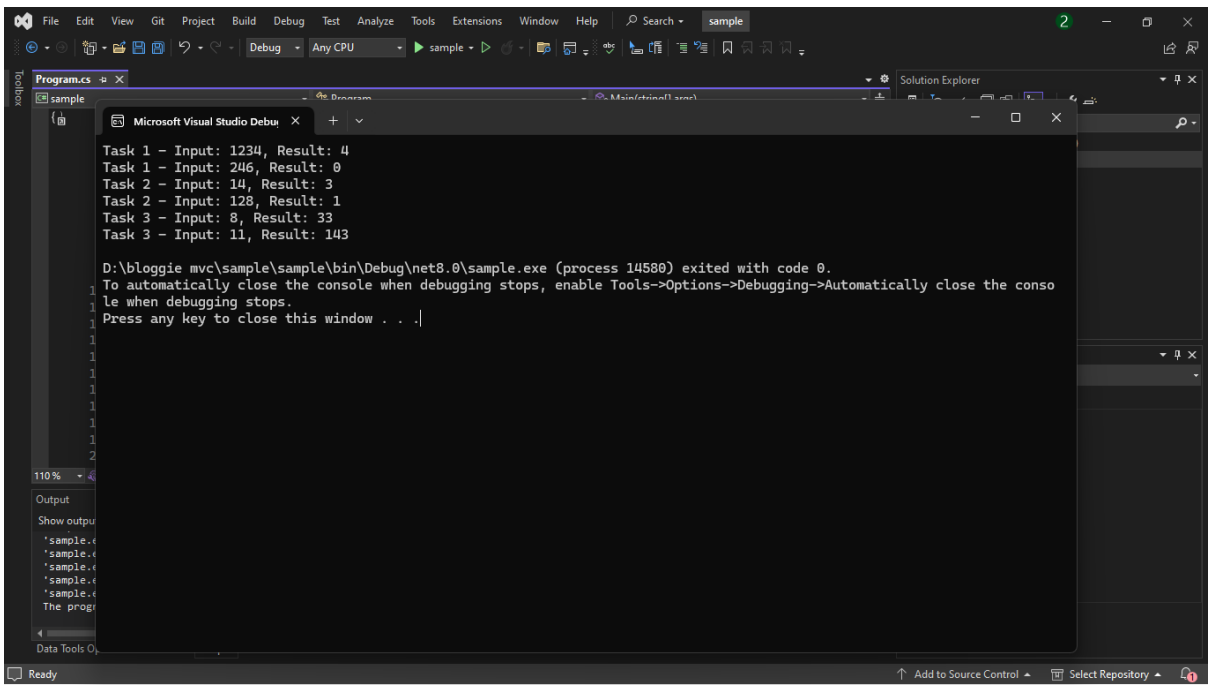
Q1) TASK1,2,3

```
using System;

class Program
{
   static void Main(string[] args)
   {
      // Task 1
      int n1 = 1234;
      int result1 = CalculateSumOfOddDigits(n1);
      Console.WriteLine($"Task 1 - Input: {n1}, Result: {result1}");

      int n2 = 246;
      int result2 = CalculateSumOfOddDigits(n2);
      Console.WriteLine($"Task 1 - Input: {n2}, Result: {result2}");

      // Task 2
      int n3 = 14;
      int result3 = CountOnesInBinary(n3);
      Console.WriteLine($"Task 2 - Input: {n3}, Result: {result3}");

      int n4 = 128;
      int result4 = CountOnesInBinary(n4);
      Console.WriteLine($"Task 2 - Input: {n4}, Result: {result4}");

      // Task 3
      int n5 = 8;
```

```
    int result5 = CalculateFibonacciSum(n5);
    Console.WriteLine($"Task 3 - Input: {n5}, Result: {result5}");

    int n6 = 11;
    int result6 = CalculateFibonacciSum(n6);
    Console.WriteLine($"Task 3 - Input: {n6}, Result: {result6}");
}

static int CalculateSumOfOddDigits(int n)
{
    int sum = 0;
    while (n > 0)
    {
        int digit = n % 10;
        if (digit % 2 != 0)
            sum += digit;
        n /= 10;
    }
    return sum;
}

static int CountOnesInBinary(int n)
{
    int count = 0;
    while (n > 0)
    {
        if ((n & 1) == 1)
            count++;
        n >>= 1;
    }
    return count;
```

```
    }

    static int CalculateFibonacciSum(int n)
    {
        int sum = 0;
        int a = 0, b = 1;
        for (int i = 0; i < n; i++)
        {
            sum += a;
            int temp = a;
            a = b;
            b = temp + b;
        }
        return sum;
    }
}
```

OUTPUT :



Q2)) TASK1,2,3

```
using System;

class Program
{
   static void Main(string[] args)
   {
      // Task 1
      int[] nums1 = { 10, 5, 3, 4 };
      SwapEvenValues(nums1);
      PrintArray(nums1);

      int[] nums2 = { 100, 2, 3, 4, 5 };
      SwapEvenValues(nums2);
      PrintArray(nums2);

      int[] nums3 = { 100, 2, 3, 45, 33, 8, 4, 54 };
      SwapEvenValues(nums3);
      PrintArray(nums3);

      // Task 2
      int[] nums4 = { 4, 100, 3, 4 };
      int result1 = CalculateDistance(nums4);
      Console.WriteLine($"Task 2 - Result: {result1}");

      int[] nums5 = { 5, 50, 50, 4, 5 };
      int result2 = CalculateDistance(nums5);
```

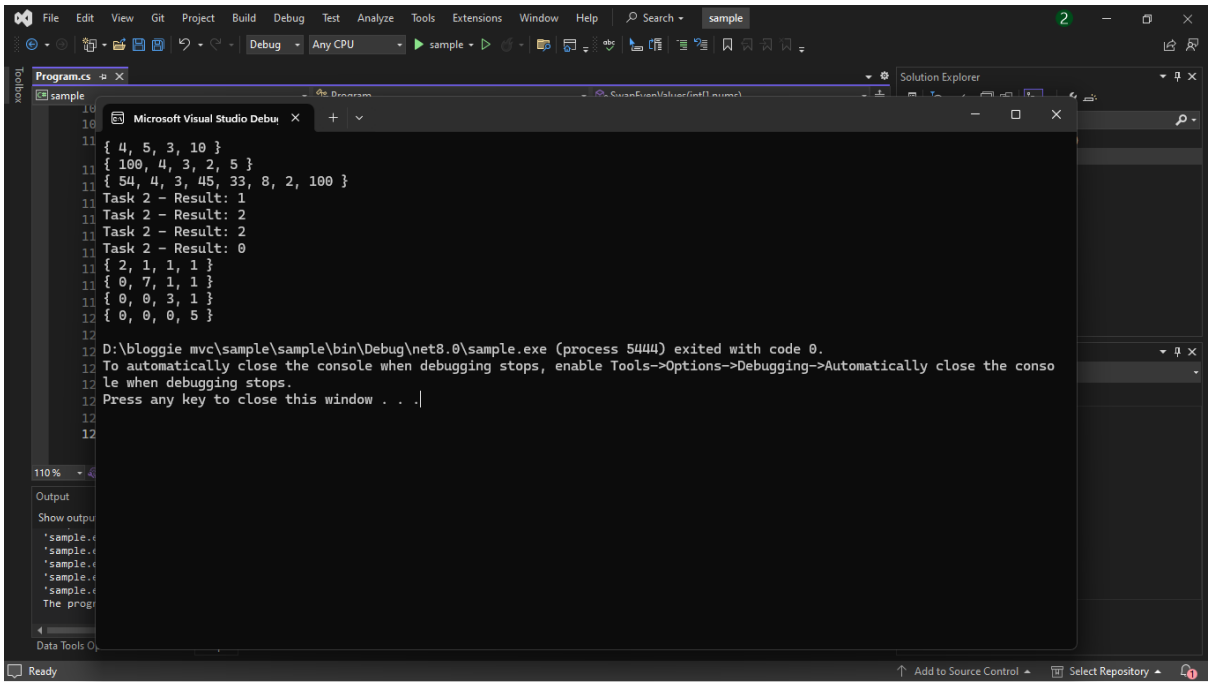
```
    Console.WriteLine($"Task 2 - Result: {result2}");

    int[] nums6 = { 5, 350, 350, 4, 350 };
    int result3 = CalculateDistance(nums6);
    Console.WriteLine($"Task 2 - Result: {result3}");

    int[] nums7 = { 10, 10, 10, 10, 10 };
    int result4 = CalculateDistance(nums7);
    Console.WriteLine($"Task 2 - Result: {result4}");

    // Task 3
    int[,] matrix = {
        { 2, 4, 3, 3 },
        { 5, 7, 8, 5 },
        { 2, 4, 3, 3 },
        { 5, 7, 8, 5 }
    };
    ModifyMatrix(matrix);
    PrintMatrix(matrix);
}

static void SwapEvenValues(int[] nums)
{
    for (int i = 0; i < nums.Length / 2; i++)
    {
        if (nums[i] % 2 == 0 && nums[nums.Length - 1 - i] % 2 == 0)
        {
            int temp = nums[i];
            nums[i] = nums[nums.Length - 1 - i];
            nums[nums.Length - 1 - i] = temp;
        }
```

```
    }
}

static int CalculateDistance(int[] nums)
{
    int maxIndex = 0;
    int minIndex = 0;
    int max = nums[0];

    for (int i = 1; i < nums.Length; i++)
    {
        if (nums[i] > max)
        {
            max = nums[i];
            maxIndex = i;
        }
        else if (nums[i] < nums[minIndex])
        {
            minIndex = i;
        }
    }

    return Math.Abs(maxIndex - minIndex);
}

static void ModifyMatrix(int[,] matrix)
{
    int n = matrix.GetLength(0);

    for (int i = 0; i < n; i++)
    {
```

```
      for (int j = 0; j < n; j++)
      {
         if (j < i)
            matrix[i, j] = 0;
         else if (j > i)
            matrix[i, j] = 1;
      }
   }
}

static void PrintArray(int[] arr)
{
   Console.Write("{ ");
   for (int i = 0; i < arr.Length; i++)
   {
      Console.Write($"{arr[i]}");
      if (i != arr.Length - 1)
         Console.Write(", ");
   }
   Console.WriteLine(" }");
}

static void PrintMatrix(int[,] matrix)
{
   int n = matrix.GetLength(0);
   for (int i = 0; i < n; i++)
   {
      Console.Write("{ ");
      for (int j = 0; j < n; j++)
      {
         Console.Write($"{matrix[i, j]}");
```

```
            if (j != n - 1)
                Console.Write(", ");
        }
        Console.WriteLine(" }");
    }
  }
}
```

OUTPUT :



Q3)) TASK1,2,3

```
using System;

public enum SortOrder
{
```

```
    Ascending,
    Descending
}

class Program
{
    static void Main(string[] args)
    {
        // Task 1
        int[] array1 = { 1, 2, 3, 4, 5 };
        Console.WriteLine($"Array is sorted in ascending order: {IsSorted(array1,
SortOrder.Ascending)}");

        int[] array2 = { 5, 4, 3, 2, 1 };
        Console.WriteLine($"Array is sorted in descending order: {IsSorted(array2,
SortOrder.Descending)}");

        // Task 2
        int[] array3 = { 5, 17, 24, 88, 33, 2 };
        Transform(array3, SortOrder.Ascending);
        PrintArray(array3);

        int[] array4 = { 15, 10, 3 };
        Transform(array4, SortOrder.Ascending);
        PrintArray(array4);

        int[] array5 = { 15, 10, 3 };
        Transform(array5, SortOrder.Descending);
        PrintArray(array5);

        // Task 3
```

```
    Console.WriteLine($"Multiplication of arithmetic progression: {MultArithmeticElements(5, 3, 4)}");

    // Task 4
    Console.WriteLine($"Sum of geometric progression elements: {SumGeometricElements(100, 0.5, 20)}");
  }

  static bool IsSorted(int[] array, SortOrder order)
  {
    if (order == SortOrder.Ascending)
    {
      for (int i = 0; i < array.Length - 1; i++)
      {
        if (array[i] > array[i + 1])
          return false;
      }
    }
    else
    {
      for (int i = 0; i < array.Length - 1; i++)
      {
        if (array[i] < array[i + 1])
          return false;
      }
    }
    return true;
  }

  static void Transform(int[] array, SortOrder order)
  {
    if (IsSorted(array, order))
```

```
    {
        for (int i = 0; i < array.Length; i++)
        {
            array[i] += i;
        }
    }
}

static double MultArithmeticElements(double a1, double t, int n)
{
    double result = 1;
    double current = a1;
    for (int i = 0; i < n; i++)
    {
        result *= current;
        current += t;
    }
    return result;
}

static double SumGeometricElements(double a1, double t, double alim)
{
    double sum = 0;
    double current = a1;
    while (current > alim)
    {
        sum += current;
        current *= t;
    }
    return sum;
}
```

```
    static void PrintArray(int[] array)
    {
        Console.Write("{ ");
        for (int i = 0; i < array.Length; i++)
        {
            Console.Write($"{array[i]}");
            if (i != array.Length - 1)
                Console.Write(", ");
        }
        Console.WriteLine(" }");
    }
}
```

OUTPUT :