

Lab 2: Tasks on Looping, Arrays and Functions

Name: Gondrala Mani Sai

Id num: 2100031545

IN-LAB:

1. Write a C# code to implement the Tasks on Looping Statements?

TASK1: For a positive integer n calculate the *result* value, which is equal to the sum of the odd numbers in n

Example

```
n = 1234    result = 4 (1 + 3)
n = 246     result = 0
```

TASK2: For a positive integer n calculate the result value, which is equal to the sum of the “1” in the binary representation of n .

Example

```
n = 14(decimal) = 1110(binary)    result = 3
n = 128(decimal) = 1000 0000(binary) result = 1
```

TASK3: For a positive integer n , calculate the result value equal to the sum of the first n Fibonacci numbers Note: Fibonacci numbers are a series of numbers in which each next number is equal to the sum of the two preceding ones: 0, 1, 1, 2, 3, 5, 8, 13... (F0=0, F1=F2=1, then F(n)=F(n-1)+F(n-2) for n>2)

Example

```
n = 8    result = 33
n = 11   result = 143
```

Solution:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Home_assign_1
{
    internal class Q1
    {
        public int CalculateSumOfOddNumbers(int n)
        {
            int sum = 0;
            while (n > 0)
            {
                int digit = n % 10;
                if (digit % 2 != 0)
                {
                    sum += digit;
                }
                n /= 10;
            }
            return sum;
        }
    }
}
```

```

    }

    public int CalculateNumberOfOnesInBinary(int n)
    {
        int count = 0;
        while (n > 0)
        {
            count += n & 1;
            n >>= 1;
        }
        return count;
    }

    public int CalculateSumOfFibonacciNumbers(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;
        }
        return sum;
    }
}

```

solution:

```

C:\WINDOWS\system32\cmd.  X  +  v

TASK 1
For n = 1234, result = 4
For n = 246, result = 0

TASK 2
For n = 14, result = 3
For n = 128, result = 1

TASK 3
For n = 8, result = 33
For n = 11, result = 143
Press any key to continue . . . |

```

2. Write a C# code to implement the Tasks on Arrays?

TASK 1: In a given array of integers *nums* swap values of the first and the last array elements, the second and the penultimate etc., if the two exchanged values are even

Example

```
{ 10 , 5, 3, 4}          => {4, 5, 3, 10}
{100, 2, 3, 4, 5}        => {100, 4, 3, 2, 5}
{100, 2, 3, 45, 33, 8, 4, 54} => {54, 4, 3, 45, 33, 8, 2, 100}
```

TASK 2: In a given array of integers *nums* calculate integer *result* value, that is equal to the distance between the first and the last entry of the maximum value in the array.

Example

```
{4, 100!, 3, 4}          result = 0
{5, 50!, 50!, 4, 5}      result = 1
{5, 350!, 350, 4, 350!} result = 3
{10!, 10, 10, 10, 10!}  result = 4
```

TASK 3: In a predetermined two-dimensional integer array (square matrix) *matrix* insert 0 into elements to the left side of the main diagonal, and 1 into elements to the right side of the diagonal.

Example

```
{{2, 4, 3, 3},      {{2, 1, 1, 1},
{5, 7, 8, 5},      => {0, 7, 1, 1},
{2, 4, 3, 3},      {0, 0, 3, 1},
{5, 7, 8, 5}}      {0, 0, 0, 5}}
```

Solution:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Home_assign_1
{
    internal class Q2
    {
        public void Task1(int[] nums)
        {
            for (int i = 0, j = nums.Length - 1; i < j; i++, j--)
            {
                if (nums[i] % 2 == 0 && nums[j] % 2 == 0)
                {
                    int temp = nums[i];
                    nums[i] = nums[j];
                    nums[j] = temp;
                }
            }
        }

        public int Task2(int[] nums)
        {
            int maxIndex = Array.IndexOf(nums, nums.Max());
            int minIndex = Array.IndexOf(nums, nums.Min());
            return Math.Abs(maxIndex - minIndex);
        }
    }
}
```

```

public void Task3(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;
            }
        }
    }
}

```

Solution:

```

TASK 1
4 5 3 10
100 4 3 2 5
54 4 3 45 33 8 2 100

TASK 2
Result = 1
Result = 2
Result = 2
Result = 0

TASK 3
2 1 1 1
0 7 1 1
0 0 3 1
0 0 0 5
Press any key to continue . . . |

```

3. Write a C# code to implement the Tasks on Functions?

TASK 1: Create function *IsSorted*, determining whether a given *array* of integer values of arbitrary length is sorted in a given *order* (the order is set up by enum value *SortOrder*). Array and sort order are passed by parameters. Function does not change the array

TASK 2: Create function *Transform*, replacing the value of each element of an integer *array* with the sum of this element value and its index, only if the given *array* is sorted in the given *order* (the order is set up by enum value *SortOrder*). Array and sort order are passed by parameters. To check, if the array is sorted, the function *IsSorted* from the Task 1 is called.

Example

```
For {5, 17, 24, 88, 33, 2} and "ascending" sort order values in the array
do not change;
For {15, 10, 3} and "ascending" sort order values in the array do not
change;
For {15, 10, 3} and "descending" sort order the values in the array change
to {15, 11, 5}
```

TASK 3: Create function *MultArithmeticElements*, which determines the multiplication of a given number of first *n* elements of arithmetic progression of real numbers with a given initial element of progression *a(1)* and progression step *t*. *a(n)* is calculated by the formula $a(n+1) = a(n) + t$.

Example

```
For a(1) = 5, t = 3, n = 4 multiplication equals to 5*8*11*14 = 6160
```

TASK 4: Create function *SumGeometricElements*, determining the sum of the first elements of a decreasing geometric progression of real numbers with a given initial element of a progression *a(1)* and a given progression step *t*, while the last element must be greater than a given *alim*. *an* is calculated by the formula $a(n+1) = a(n) * t$, $0 < t < 1$.

Example

```
For a progression, where a(1) = 100, and t = 0.5, the sum of the first
elements, grater than alim = 20, equals to 100+50+25 = 175
```

Solution:

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Home_assign_1
{
```

```

internal class Q3
{
    public static bool IsSorted(int[] arr, SortOrder order)
    {
        if (order == SortOrder.Ascending)
        {
            for (int i = 0; i < arr.Length - 1; i++)
            {
                if (arr[i] > arr[i + 1])
                    return false;
            }
        }
        else if (order == SortOrder.Descending)
        {
            for (int i = 0; i < arr.Length - 1; i++)
            {
                if (arr[i] < arr[i + 1])
                    return false;
            }
        }
        return true;
    }

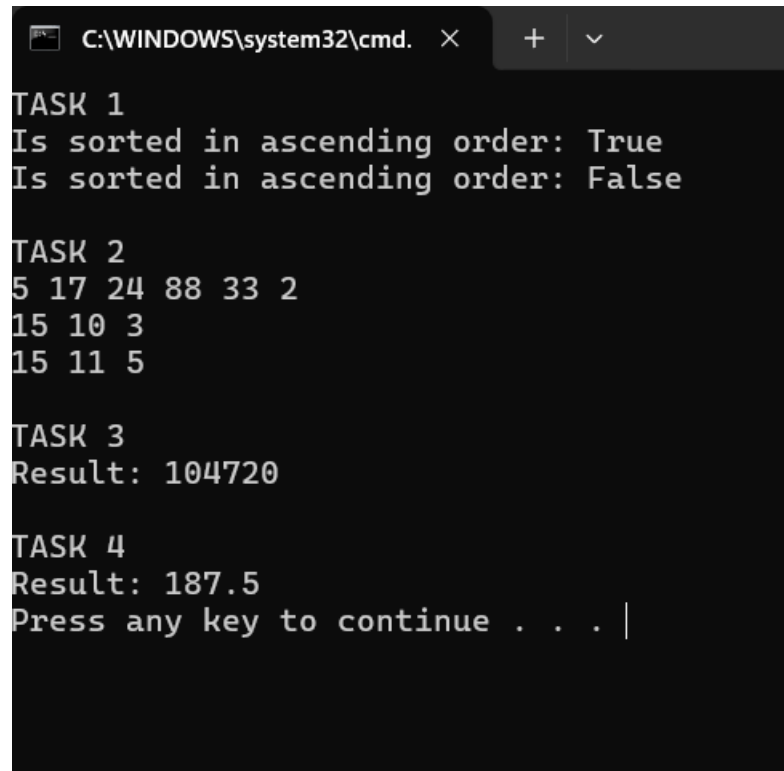
    public static void Transform(int[] arr, SortOrder order)
    {
        if (IsSorted(arr, order))
        {
            for (int i = 0; i < arr.Length; i++)
            {
                arr[i] += i;
            }
        }
    }

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

    public static double SumGeometricElements(double a1, double t, double
alim)
    {
        double sum = a1;
        double nextTerm = a1;
        while (nextTerm > alim)
        {
            nextTerm *= t;
            sum += nextTerm;
        }
        return sum;
    }
}

```

Solution



```
C:\WINDOWS\system32\cmd.  X  +  v

TASK 1
Is sorted in ascending order: True
Is sorted in ascending order: False

TASK 2
5 17 24 88 33 2
15 10 3
15 11 5

TASK 3
Result: 104720

TASK 4
Result: 187.5
Press any key to continue . . . |
```



