Assignment 2

2.5

1. Reverse the ordering of words in a String

I have this string

s1 = "My name is X Y Z"

and I want to reverse the order of the words so that

s1 = "Z Y X is name My"

Input;

using System;

static class StringHelper

{

public static string ReverseString(string myStr)

{

char[] myArr = myStr.ToCharArray();

Array.Reverse(myArr);

return new string(myArr);

}

}

class myProgram

{

static void Main()

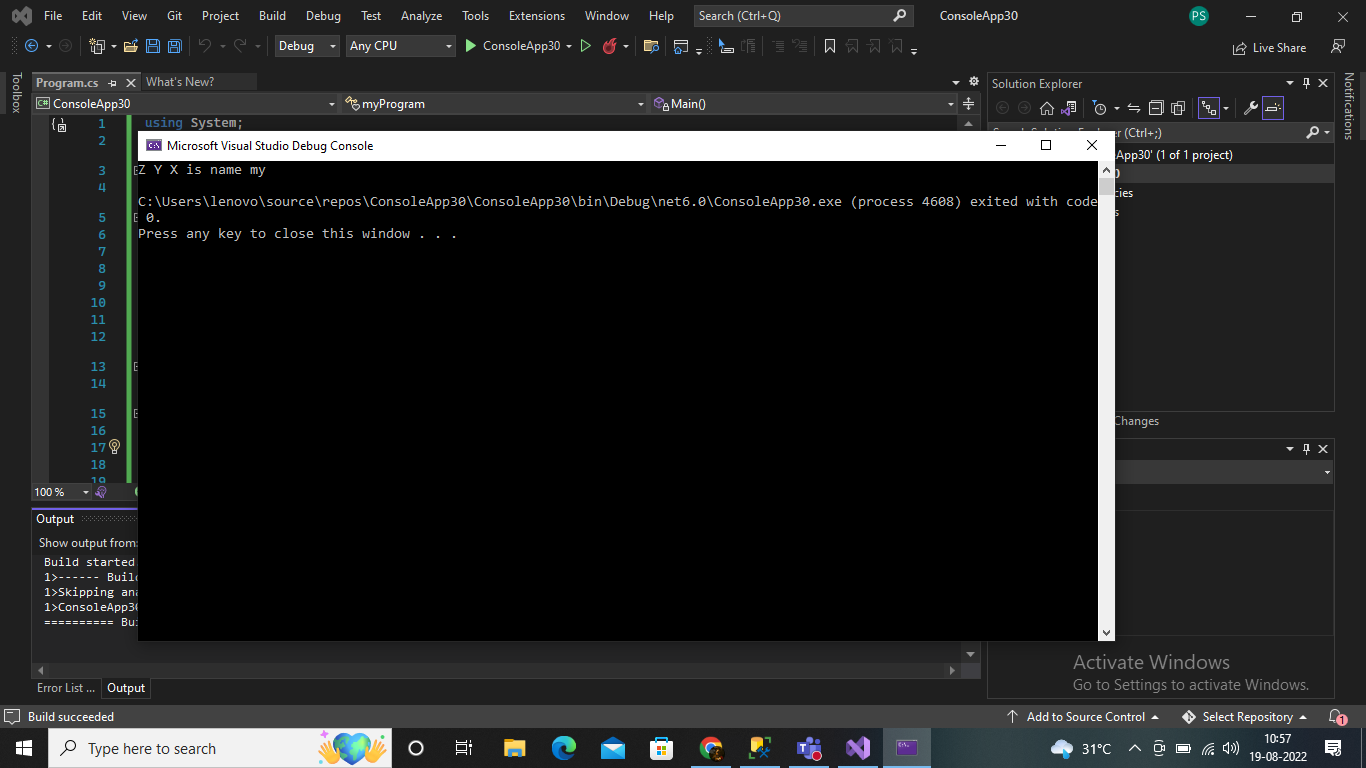
{

Console.WriteLine(StringHelper.ReverseString("ym eman si X Y Z"));// pass the desired string

}

}

Output



1. q.3 Write a function:

class Solution { public int solution(int[] A); }

that, given an array A of N integers, returns the smallest positive integer (greater than 0) that does not occur in A.

For example, given A = [1, 3, 6, 4, 1, 2], the function should return 5.

Given A = [1, 2, 3], the function should return 4.

Given A = [−1, −3], the function should return 1.

Assume that:

N is an integer within the range [1..100,000]; each element of array A is an integer within the range [−1,000,000..1,000,000]. Complexity:

Input;

using System;

using System.Linq;

using System.Collections.Generic;

public class Program

{

static void Main()

{

int[] A = { 1, 2, 3 };

var smallestInt = 1;

smallestInt = GetSmallestPositiveInteger(A);

Console.WriteLine(smallestInt);

}

public static int solution(int[] array)

{

HashSet<int> found = new HashSet<int>();

for (int i = 0; i < array.Length; i++)

{

if (array[i] > 0)

{

found.Add(array[i]);

}

}

int result = 1;

while (found.Contains(result))

{

result++;

}

return result;

}

public static int solution1(int[] A)

{

int flag = 1;

A = A.OrderBy(x => x).ToArray();

for (int i = 0; i < A.Length; i++)

{

if (A[i] <= 0)

continue;

else if (A[i] == flag)

{

flag++;

}

}

return flag;

}

public static int GetSmallestPositiveInteger(int[] A)

{

var smallestInt = 1;

var arrLn = A.Length;

if ((A == null) || (A.Length == 0)) { return 1; }

return arrLn;

A = A.OrderBy(x => x).ToArray();

for (int i = 0; i < A.Length; i++)

{

if (A[i] <= 0)

continue;

else if (A[i] == smallestInt)

{

smallestInt++;

}

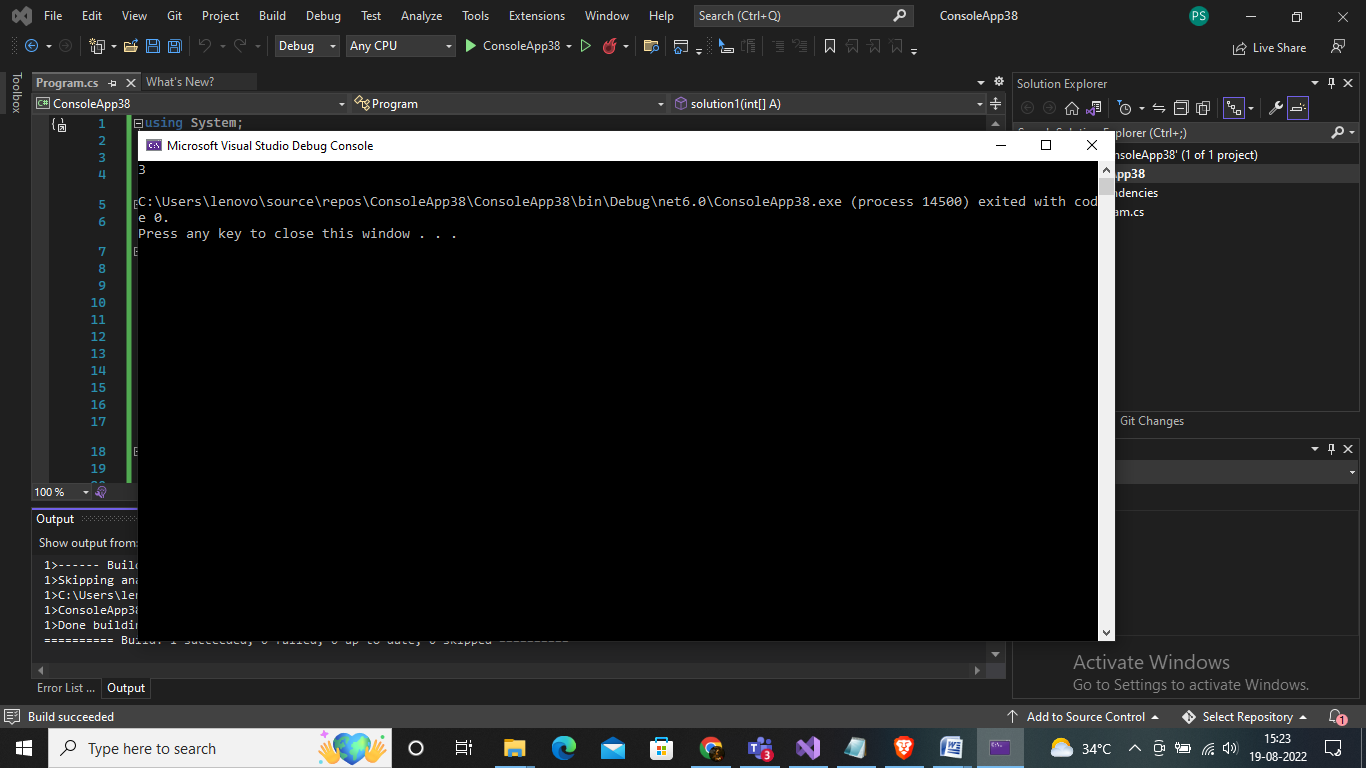
}

return smallestInt;

}

}

Output;



4. Given six digits, find the earliest valid time that can be displayed on a digital clock (in 24-hour format) using those digits.

For example, given digits 1, 8, 3, 2, 6, 4 the earliest valid time is "12:36:48". Note that "12 : 34 : 68" is not a valid time.

Write a method:

class Solution { public String solution(int A, int B, int C, int D, int E, int F); }

That, given six integers A, B, C, D, E and F, returns the earliest valid time in "hh:mm:ss" string format, or "NOT POSSIBLE" if it is not possible to display a valid time using all six integers.

For example, given 1, 8, 3, 2, 6, 4 the function should return "12:36:48".

Given 0, 0, 0, 0, 0, 0, the function should return "00:00:00". Given 0, 0, 0, 7, 8, 9, the function should return "07:08:09". Given 2, 4, 5, 9, 5, 9, the function should return "NOT POSSIBLE".

Assume that: • A, B, C, D, E and F are integers within the range [0..9].

Solution;

public string solution(int A, int B, int C, int D, int E, int F)

{

int temp = 0; Double testNum = 0.0;

int[] numsArray = new int[] { A, B, C, D, E, F };

Arrays.sort(numsArray);

for (int i = 0; i < 6; i++)

{

if (numsArray[i] > 5)

temp++;

testNum = testNum + numsArray[i] \* Math.pow(10.0, 5.0 - i);

}

if (temp > 3)

{

return null;

}

if (testNum > 235959)

{

return null;

}

if (numsArray[4] > 5)

{

temp = numsArray[4];

numsArray[4] = numsArray[2];

numsArray[2] = temp;

}

if (numsArray[2] > 5)

{

temp = numsArray[2];

numsArray[2] = numsArray[3];

numsArray[3] = temp;

}

if (numsArray[2] > 5)

{

temp = numsArray[1];

numsArray[1] = numsArray[2];

numsArray[2] = temp;

}

if (numsArray[2] > numsArray[4])

{

temp = numsArray[4];

numsArray[4] = numsArray[2];

numsArray[2] = temp;

}

return ("" + numsArray[0] + numsArray[1] + ":" + numsArray[2] + numsArray[3] + ":" + numsArray[4] + numsArray[5]);

}

1. Q.1 Write the C# code to display “<n> minute(s) ago" if difference between both dates is less than an hour and “<n> hour(s) ago" if the difference is less than a day and “<n> day(s) ago" if the difference is less than a month and the date if the difference is more than a month.

using System;

using System.Threading;

using System.Diagnostics;

public class Demo

{

public static void Main()

{

DateTime date1 = new DateTime(1999, 1, 13, 3, 57, 32, 11);

DateTime date2 = new DateTime(1999, 1, 14, 3, 59, 32, 11);

TimeSpan t = date2.Subtract(date1);

double min = t.TotalMinutes;

double hr = min / 60;

double day = t.TotalDays;

double month = day / 30;

double year = month / 12;

Console.WriteLine(t.TotalMinutes);

if (day >= 1 && day < 30)

{

Console.WriteLine("Days (Difference) = {0} ", Math.Floor(t.TotalDays\*100)/100 + " day ago");

}

else if (min>=1 && min < 59)

{

Console.WriteLine("Minutes (Difference) = {0}", t.TotalMinutes + " min ago");

}

else if (hr >= 1 && hr < 24)

{

Console.WriteLine("hr (Difference) = {0}", Math.Floor(hr \* 100) / 100 + " hr ago");

}

else if (month >= 1 && month <12)

{

Console.WriteLine("Month (Difference) = {0}", Math.Floor(month \* 100) / 100 + " month ago");

}

else if (year >= 1)

{

Console.WriteLine("Year (Difference) = {0}", Math.Floor(year \* 100) / 100 + " year ago");

}

}

}

Output

