1. **SORT THE LIST**

Write a program which reads an Integer(size of the list), a String List and a character, and to get the strings that will not start with the given character irrespective of case. Sort the elements in ascending order based on its length. Print the output list. If the elements are having the same length, then display the elements in alphabetical order.  
 Include a class UserProgramCode with static method  getTheElements which accepts a String list  and a character. The return type is List<String>.  
 Create a Class Program which would be used to get the inputs and call the static method present in UserProgramCode. In getTheElements method,  return "Invalid Input" when any of the input strings contain non-alphabets. When the output list is empty, then return "List is Empty". Otherwise return the appropriate result.  
 In Program class print the result which is returned by getTheElements method in UserProgramCode. Input output format The first line of the input is an integer that corresponds to n, the size of the list. The next n lines of input correspond to the elements in the string list. The next line of the input contains the  character. Sample Input 1: 3 read write edit e Sample Output 1: read write Sample Input2 : 2 Elegent event e Sample Output2 : List is Empty Sample Input 3: 2 Eleg$ent e^ent e Sample Output 3 : Invalid Input

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace new\_practice**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**int n = int.Parse(Console.ReadLine());**

**List<string> list = new List<string>();**

**for (int i = 0; i < n; i++)**

**{**

**list.Add(Console.ReadLine());**

**}**

**char c = char.Parse(Console.ReadLine());**

**List<string> op = UserProgramCode.Sort\_string(list, c);**

**if (list[0] == "-1")**

**{**

**Console.WriteLine("List is empty");**

**}**

**else if (list[0] == "-2")**

**{**

**Console.WriteLine("Invalid Input");**

**}**

**else**

**for (int i = 0; i < op.Count; i++)**

**{**

**Console.WriteLine(op[i]);**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace new\_practice**

**{**

**class UserProgramCode**

**{**

**public static List<string> Sort\_string(List<string> list, Char c)**

**{**

**string temp = "";**

**for (int i = 0; i < list.Count; i++)**

**{**

**if (list[i].StartsWith(c.ToString()))**

**{**

**list.Remove(list[i]);**

**i--;**

**}**

**}**

**if (list.Count == 0)**

**{**

**list.Add("-1");**

**return list;**

**//Console.WriteLine("List is empty");**

**//Environment.Exit(0);**

**}**

**for (int i = 0; i < list.Count; i++)**

**{**

**for (int j = 0; j < list[i].Length; j++)**

**{**

**if (!char.IsLetter(list[i][j]))**

**{**

**list.Clear();**

**list.Add("-2");**

**return list;**

**//Console.WriteLine("Invalid Input");**

**//Environment.Exit(0);**

**}**

**}**

**}**

**for (int i = 0; i < list.Count - 1; i++)**

**{**

**for (int j = i + 1; j < list.Count; j++)**

**{**

**if (list[i].Length > list[j].Length)**

**{**

**temp = list[i];**

**list[i] = list[j];**

**list[j] = temp;**

**}**

**else if(list[i].Length==list[j].Length)**

**{**

**if (list[i][0] > list[j][0])**

**{**

**temp = list[i];**

**list[i] = list[j];**

**list[j] = temp;**

**}**

**}**

**}**

**}**

**//for (int i = 0; i < list.Count - 1; i++)**

**//{**

**// for (int j = i + 1; j < list.Count; j++)**

**// {**

**// if (list[i][0] > list[j][0])**

**// {**

**// temp = list[i];**

**// list[i] = list[j];**

**// list[j] = temp;**

**// }**

**// }**

**//}**

**return list;**

**}**

**}**

**}**

1. IDENTIFY PERFECT NUMBERS.

For a given integer input array ,write a program to identify the perfect numbers in the input array and store remaining elements excluding the perfect numbers in the output .Perfect number is a positive number in which the sum of all its positive divisors excluding that number is equivalent to that number itself. Eg. 6 is a perfect number ,since its divisor are 1, 2 and 3. Sum of its divisors is 1 + 2+ 3 = 6,which is equal to the number itself. Business rule: 1) If any of the elements in input1 array is negative, then print -1. 2) If there are any duplicates found in input1 array, then print -2. 3) If size of the input1 array is 1 or greater than 7, then print -3. Create a class named UserProgramCode that has the following static method   
 public static int[] perfectNum(int[] input1)  
 Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.    
 Input and Output Format:  
 The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.  
 The next 'n' lines of input consist of elements in the input array.  
 Refer business rules and sample output for formatting specifications. Sample Input 1 :  
 4 6 2 5 7  
 Sample Output 1 : 2 5 7 Sample Input 2 : 5 5 8 3 -4 6    
 Sample Output 2 : -1

**identify perfect**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace perfectnumber**

**{**

**class UserProgramCode**

**{**

**public static int[] perfect(int[] arr,int n)**

**{**

**int sum = 0;**

**List<int> temp = new List<int>();**

**List<int> temp1 = new List<int>();**

**if (arr.Length < 1 || arr.Length > 7)**

**{**

**temp1.Add(-3);**

**return temp1.ToArray();**

**}**

**for (int i = 0; i < n; i++)**

**{**

**for (int j = 1; j < n; j++)**

**{**

**if (arr[i] == arr[j])**

**{**

**temp1.Add(-2);**

**return temp1.ToArray();**

**}**

**}**

**}**

**for (int i = 0; i < n; i++)**

**{**

**sum = 0;**

**if (arr[i] < 0)**

**{**

**temp1.Add(-1);**

**return temp1.ToArray();**

**}**

**for (int j = 1; j <= (arr[i] / 2); j++)**

**{**

**if (arr[i] % j == 0)**

**{**

**sum = sum + j;**

**}**

**}**

**if (sum != arr[i])**

**{**

**temp.Add(arr[i]);**

**}**

**}**

**return temp.ToArray();**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace perfectnumber**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**int n = int.Parse(Console.ReadLine());**

**int[] arr= new int[n];**

**for(int i=0;i<n;i++)**

**{**

**arr[i]=int.Parse(Console.ReadLine());**

**}**

**int[] op = UserProgramCode.perfect(arr, n);**

**foreach (int item in op)**

**{**

**Console.WriteLine(item);**

**}**

**Console.ReadLine();**

**}**

**}**

}

**93.PASSWORD VALIDATION**

Password Validation  
 Given a method with a password in string format as input,  write code to validate the password using following rules:  
    
 - Must contain at least one digit  
 - Must contain at least one of the following special characters @, #, $  
 - Length should be between 6 and 20 characters (both inclusive).  
    
 Include a class UserProgramCode with a static method validatePassword which accepts a password string as input.  
 If the password is as per the given rules return 1 else return -1.If the return value is 1 then print "Valid password" else print as "Invalid password".  
 Create a Program class which gets a string as an input and call the static method validatePassword present in the UserProgramCode.  
    
 Input and Output Format:  
 Input is a string .  
 Output consists of a string. Output "Valid password" if the given password is valid or "Invalid password" if the given password is not valid.  
 Sample Input 1:  
 %Dhoom%  
 Sample Output 1: Invalid password  
    
 Sample Input 2:  
 #@6Don  
 Sample Output 2:  
 Valid password

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace level1\_56**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**string str1;**

**int x;**

**str1 = Console.ReadLine();**

**x=UserProgramCode.validatePassword(str1);**

**if(x==1)**

**Console.WriteLine("valid input");**

**else**

**Console.WriteLine("Invalid input");**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace level1\_56**

**{**

**class UserProgramCode**

**{**

**public static int validatePassword(string str)**

**{**

**bool a, b, c;**

**a = str.Contains("@");**

**b = str.Contains("#");**

**c = str.Contains("$");**

**if (a || b ||s c)**

**{**

**if ((str.Length >= 6) && (str.Length <= 20))**

**{**

**return 1;**

**}**

**}**

**return -1;**

**}**

**}**

**}**

**94.COUNT EVEN OCCURRENCE**

Count Even Occurrence  
    
 Given an int array, write a program to calculate the count as follows. If the same element is repeated even number of times,increase the count by one. Print the value of the count. Business Rules : 1. If any of the element in the array is a negative number, then print -1. 2. If there is no elements repeated in even number of times, then print 0. Create a class named UserProgramCode that has the following static method   
 public static int countEvenOccurence(int[] input1)  
 Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.    
 Input and Output Format:  
 The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.  
 The next 'n' lines of input consist of elements in the input array.  
 Output consists of an integer.  
 Refer sample output for formatting specifications.    
 Sample Input 1 : 17 1 2 3 4 9 3 6 7 1 9 100 2 4 1 45 1 9    
 Sample Output 1 : 4 Sample Input 2 :  
 13 1 2 3 4 3 6 7 1 9 100 2 4 -17    
 Sample Output 2 : -1

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace CountEvenOccurance**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**int n = int.Parse(Console.ReadLine());**

**int[] arr = new int[n];**

**for (int i = 0; i < n; i++)**

**{**

**arr[i] = int.Parse(Console.ReadLine());**

**}**

**int op = UserProgramCode.CountEvenOccurance(arr);**

**Console.WriteLine(op);**

**Console.ReadLine();**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace CountEvenOccurance**

**{**

**class UserProgramCode**

**{**

**public static int CountEvenOccurance(int[] arr)**

**{**

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

**{**

**if (arr[i] < 0)**

**return -1;**

**}**

**int c=0;**

**int final=0;**

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

**{**

**c = 1;**

**for(int j=1;j<arr.Length;j++)**

**{**

**if(arr[i]==arr[j])**

**{**

**c++;**

**}**

**}**

**if (c % 2 == 0)**

**{**

**final++;**

**}**

**else**

**return 0;**

**}**

**return final;**

**}**

**}**

**}**

**95.ARRAY MEDIAN**

Given an integer array as input, write a program to calculate median of the given numbers in the array with the below conditions 1. Sort the Input array in ascending. 2. When the number of elements in the Input array is odd, then median will be the middle number. Median is the (N+1)/2th element. 3. When the number of elements in the Input array is even, then median will be the average of two middle numbers. Round off the median value to the nearest integer. Business rule: 1) If the Input array contains any negative numbers, then print -1. 2) If any of the Input array elements contains "0", then print -2. Example1: Input : 7 1 2 1 4 7 1 2 Output : 2 After sorting the array is {1,1,1,2,2,4,7} Number of element in input array N is 7 (N+1)/2th element= (7+1)/2= 4th element Median value is the 4th number which is 2 Example 2: Input : 6 52 51 81 84 60 88 Output : 71 After sorting the array is {51,52,60, 81, 84, 88} Median = Average of Middle 2 numbers = (60 + 81)/2 = 70.5. round(70.5) = 71    
 Create a class named UserProgramCode that has the following static method   
 public static int calculateMedian(int[] input1) Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.    
 Input and Output Format:  
 The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.  
 The next 'n' lines of input consist of elements in the input array.  
 Refer business rules and sample output for formatting specifications.

Sample Input :

7 1 2 1 4 7 1 2  
 Sample Output :

2

**------Arrray Median----**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace Arraymedian**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**int x, i;**

**double m;**

**x =Convert.ToInt32( Console.ReadLine());**

**int[] input = new int[x];**

**for (i = 0; i < x; i++)**

**{**

**input[i]=Convert.ToInt32(Console.ReadLine());**

**}**

**median c = new median();**

**m=c.arraymedian(input);**

**double d=Math.Round(m,0,MidpointRounding.AwayFromZero);**

**Console.WriteLine(d);**

**Console.ReadLine();**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace Arraymedian**

**{**

**class median**

**{**

**public double arraymedian(int[] input)**

**{**

**int length;**

**double p;**

**Array.Sort(input);**

**input.Reverse();**

**length = input.Length;**

**foreach (int i in input)**

**{**

**if (i < 0)**

**{**

**return -1;**

**}**

**if (i == 0)**

**{**

**return -2;**

**}**

**}**

**if (length % 2 == 1)**

**{**

**return input[((length + 1) / 2)-1];**

**}**

**else**

**{**

**p= (float)(input[(length / 2)-1] + input[(length / 2)])/2.0;**

**return p;**

**}**

**}**

**}**

**}**

**96.REVERSING A STRING**

Reversing a String    
 Given a method that accepts a string and a Character as its input parameters, Write code to reverse the string and return it in a format such that each character is separated by the given character. Create a class named UserProgramCode that has the following static method   
 public static string reshape(string input1, char input2)   
 Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.    
 Input and Output Format:  
 Input consists of a string and a character.  
 Output is a string.  
    
 Sample Input :  
 Rabbit  
 -  
    
 Sample Output :  
 t-i-b-b-a-R

**using System;**

**do this first**

**class Program**

**{**

**public static void Main( string[] args )**

**{**

**string str=Console.ReadLine();**

**char ch = Convert.ToChar(Console.ReadLine());**

**Console.WriteLine(UserProgramCode.reshape(str,ch));**

**Console.Read();**

**}**

**}**

**-----------------------**

**USER PROGRAM**

**-----------------------**

**using System;**

**class UserProgramCode**

**{**

**public static string reshape(string str, char ch)**

**{**

**int l = str.Length;**

**string sree = "";**

**char[] temp = str.ToCharArray();**

**for (int i = l - 1; i >= 0; i--)**

**{**

**sree = string.Concat(sree, temp[i]);**

**if (i != 0)**

**{**

**sree = string.Concat(sree, ch);**

**}**

**}**

**return (sree);**

**}**

**}**

**till here**

**else down look**

**===================**

**using System;**

**class Program**

**{**

**public static void Main( string[] args )**

**{**

**string inputWord=Console.ReadLine();**

**int position=Convert.ToInt32(Console.ReadLine());**

**char ch=Convert.ToChar(Console.ReadLine());**

**string result=UserProgramCode.replaceString(inputWord,position,ch);**

**if(result.Equals("-1"))**

**Console.WriteLine("Invalid String");**

**else if(result.Equals("-2"))**

**Console.WriteLine("Number not positive");**

**else if(result.Equals("-3"))**

**Console.WriteLine("Character not a special character");**

**else**

**Console.WriteLine(result);**

**}**

**}**

**using System;**

**class UserProgramCode**

**{**

**public static string replaceString(string inputWord, int position, char ch)**

**{**

**string inputWord1=inputWord.ToLower();**

**foreach (Char z in inputWord)**

**{**

**if (!(Char.IsLetterOrDigit(z) || Char.IsWhiteSpace(z)))**

**{**

**return "-1";**

**}**

**}**

**if (position <= 0)**

**{**

**return "-2";**

**}**

**if ((Char.IsLetterOrDigit(ch)) || Char.IsWhiteSpace(ch))**

**{**

**return "-3";**

**}**

**else**

**{**

**string[] A = inputWord1.Split(' ');**

**string b = string.Copy(A[position - 1]);**

**char[] B = b.ToCharArray();**

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

**{**

**B[i] = ch;**

**}**

**string c = new string(B);**

**A[position - 1] = c;**

**string d = string.Join(" ", A);**

**return d;**

**}**

**}**

**}**

**97.LARGEST SPAN**

Write a program to read the size of the integer array and the elements of the array and find the size of largest Span in the given array. Print the output.  
    
 Note: Span is the number of elements between two repeated numbers including both numbers. Assume an array with single element has a span of 1.   
 Business rule:   
 If there is no number repeated in an array, return 0. If there are two repeated integers in the input array, consider the first number and return the span.   
    
 Example 1:   
 Input = {1, 2, 1, 1, 3}   
 Output = 4   
 Example 2:   
 Input = {1, 4, 2, 1, 4, 1, 5}   
 Output = 6  
    
 Include a class UserProgramCode with a static method getMaxSpan which accepts the size of the array and the integer array. The return type (integer) should return the span size.  
 Create a Class Program which would be used to accept the size of the array and the array elements and call the static method present in UserProgramCode.  
 Input and Output Format:  
 Input consists of n+1 integers, where the first integer corresponds to the size of the array followed by n integers .  
 Output consists of an integer(the span size).  
 Refer sample output for formatting specifications.  
    
 Sample Input 1:  
 5  
 1  
 2  
 1  
 1  
 3  
 Sample Output 1:  
 4  
    
 Sample Input 2:  
 7  
 1  
 4  
 2  
 1  
 4  
 1  
 5  
 Sample Output 2:  
 6

NO ANSWER FOUND:::

**98. ROMAN NUMERAL**

Given an integer as input, write a program to convert integer input to roman numerals . Represent it as a string. Basic Steps for Roman number calculation: 1. I is the numeral one. V is the numeral 5. X is the numeral 10. L is the numeral 50. C is the numeral 100. D is the numeral 500. M is the numeral 1000.  
 2.A smaller number in front of a larger number means subtraction, everything else means addition. For example, IV means 4, VI means 6. You would not put more than one smaller number in front of a larger number to subtract. For example, IIV would not mean 3. You must separate ones, tens, hundreds, and thousands as separate items. This means that 99 is XCIX, 90 + 9, but never should be written as IC. Similarly, 999 cannot be IM and 1999 cannot be MIM. So, II is two, III is three. VII is 7, VIII is 8. IX is 9, XI is 11, etc. Again, XL would be 40, LX would be 60, LXX would be 70, LXXX would be 80 etc. Similarly, XC would be 90, XCIX would be 99, CL would be 150, CLIX would be 159, CXC would be 190, CC would be 200, CCC would be 300, etc. Again, CD would be 400, DC would be 600, etc. And CM would be 900. Business Rule: 1. If the input is less than 0, then print “Invalid Input” . 2. If the given input variable is greater than 4000, then print “Greater Than 4000” . Create a class named UserProgramCode that has the following static method   
 public static string romanNumerals(int input1).  
 Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.    
 Input and Output Format:  
 Input consists of an integer.  
 Output is a string.  
 Refer business rules and sample output for formatting specifications.

Sample Input 1 :  
 2086    
Sample Output 1 :

MMLXXXVI

Sample Input 2 :

2091

Sample Output 2 :

MMXCI

Sample Input 3 :

-2091    
 Sample Output 3 :  
 Invalid Input

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace numer**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**int num = int.Parse(Console.ReadLine());**

**string output =UserProgramCode.ToRoman(num);**

**Console.WriteLine(output);**

**Console.ReadLine();**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace numer**

**{**

**class UserProgramCode**

**{**

**public static string ToRoman(int num)**

**{**

**if (num > 3999) throw new ArgumentException("Too big - can't exceed 3999");**

**if (num < 1) throw new ArgumentException("Too small - can't be less than 1");**

**int thousands, hundreds, tens, units;**

**thousands = num / 1000;**

**num %= 1000;**

**hundreds = num / 100;**

**num %= 100;**

**tens = num / 10;**

**units = num % 10;**

**var sb = new StringBuilder();**

**if (thousands > 0) sb.Append(roman1[3 - thousands]);**

**if (hundreds > 0) sb.Append(roman2[9 - hundreds]);**

**if (tens > 0) sb.Append(roman3[9 - tens]);**

**if (units > 0) sb.Append(roman4[9 - units]);**

**return sb.ToString();**

**}**

**static string[] roman1 = { "MMM", "MM", "M" };**

**static string[] roman2 = { "CM", "DCCC", "DCC", "DC", "D", "CD", "CCC", "CC", "C" };**

**static string[] roman3 = { "XC", "LXXX", "LXX", "LX", "L", "XL", "XXX", "XX", "X" };**

**static string[] roman4 = { "IX", "VIII", "VII", "VI", "V", "IV", "III", "II", "I" };**

**}**

**}**

**99.DIFFERENCE BETWEEN DATES IN MONTHS**

Find the difference between Dates in months  
 Given a method with two date strings in yyyy-mm-dd format as input, write code to find the difference between two dates in months.   
    
 Include a class UserProgramCode with a static method getMonthDifference which accepts two date strings as input. The method returns an integer which is the difference between two dates in months.  
    
 Create a class Program which would get the input and call the static method getMonthDifference present in the UserProgramCode.  
    
 Input and Output Format:  
 Input consists of two date strings.  
 Format of date : yyyy-mm-dd.  
    
 Output is an integer.   
 Refer sample output for formatting specifications.  
    
 Sample Input 1:  
 2012-03-01  
 2012-04-16  
 Sample Output 1:  
 1  
    
 Sample Input 2:  
 2011-03-01  
 2012-04-16  
 Sample Output 2:  
 13

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace level1\_58**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**int k;**

**string intime, outtime;**

**intime = Console.ReadLine();**

**outtime = Console.ReadLine();**

**k = UserProgramCode.getMonthDifference(intime, outtime);**

**if (k == -1)**

**{**

**Console.WriteLine("Invalid format");**

**}**

**else**

**Console.WriteLine(k);**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace level1\_58**

**{**

**class UserProgramCode**

**{**

**public static int getMonthDifference(string intime, string outtime)**

**{**

**string s;**

**int d1 = 0;**

**s = "yyyy-MM-dd";**

**DateTime i, o;**

**bool k = DateTime.TryParseExact(intime, s, null, System.Globalization.DateTimeStyles.None, out i);**

**if (k == false)**

**return -1;**

**bool j = DateTime.TryParseExact(outtime, s, null, System.Globalization.DateTimeStyles.None, out o);**

**if (j == false)**

**return -1;**

**int a1 = i.Month;**

**int a2 = o.Month;**

**int d;**

**if (a1 > a2)**

**d = a1 - a2;**

**else**

**d = a2 - a1;**

**int y1 = i.Year;**

**int y2 = o.Year;**

**if (y1 > y2)**

**d1 = y1 - y2;**

**else**

**d1 = y2 - y1;**

**return ((d1\*12)+d);**

**}**

**}**

**}**

**100.PRINT CAPITALIZED**

Write a code to convert the first letter of each word to capital Case and return the final string Example : Input: ""Now is the time to act!"" Output: ""Now Is The Time To Act!"" Include a Class UserProgramCode with a static method printCapitalized which accepts a string as an input.The return type is String which is a sentence with first letter of each word capitalized. Create a Class Program which would be used to accept String and call the static method present in UserProgramCode. Input and Output Format: Input consists of string. Output consists of a string which corressponds to first letter of each word to be capitalized and make other letters to be lower case

Sample Input 1:

Features Of JAVA2

Sample Output 1:

Features Of Java2

Sample Input 2:

gOogLe is A SeaRCh enGinEe

Sample Output 2:

Google Is A Search Enginee

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace ConsoleApplication2**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**string input,output;**

**input = Console.ReadLine();**

**Class1 c = new Class1();**

**output=c.capitals(input);**

**Console.WriteLine(output);**

**Console.ReadLine();**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace ConsoleApplication2**

**{**

**class Class1**

**{**

**public string capitals(string input)**

**{**

**string output1;**

**char c;**

**int i = 0,k=0;**

**string[] s = new string[10];**

**char[] ch = new char[10];**

**string sout;**

**StringBuilder sb = new StringBuilder();**

**s = input.Split(' ');**

**foreach (string str in s)**

**{**

**c = str[0];**

**ch = str.ToCharArray();**

**k = 0;**

**foreach (char d in ch)**

**{**

**ch[k]=char.ToLower(d);**

**k = k + 1;**

**}**

**ch[0] = char.ToUpper(c);**

**sout = new string(ch);**

**sb.Append(sout);**

**sb.Append(' ');**

**i = i + 1;**

**}**

**return sb.ToString();**

**}**

**}**

**}**

**101.NUMBER AVAILABILITY**

Write the program to find whether the given number is available in a list of numbers.  
 Get three input parameters, one the size of the list, second the list of numbers and the other the given number to be searched. Print the output as - “Non Positive”, “Present”, or “Not Present” respectively as per the given business rules.  
    
 Business Rules:  
 1.List of numbers and the number to be searched, all of them should be positive numbers only, if not return -1.  
 2. If the given number is present in the list of numbers , then return 1.  
 3. If the given number is not present in the list of numbers , then return 0.  
    
 Include a class UserProgramCode with a static method findExistence which accepts the size of the integer array,  an integer array and the number to be searched. The return type (Integer) should return -1, 1 or 0 as per the given business rules.  
 Create a Class Program which would be used to accept the size of the array, the array elements and an integer, and call the static method present in UserProgramCode.  
 Input and Output Format:  
 Input consists of an integer  which corresponds to the size of the array, an integer array and an integer.  
 Output consists of a String(“Non Positive”, “Present”, or “Not Present” ).  
 Refer sample output for formatting specifications.  
    
 Sample Input 1:  
 3  
 1  
 2  
 3  
 1  
 Sample Output 1:  
 Present  
    
 Sample Input 2:  
 3  
 -1  
 2  
 3  
 3  
 Sample Output 2:  
 Non Positive  
    
 Sample Input 3:  
 3  
 1  
 2  
 3  
 4  
 Sample Output 3:  
 Not Present

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication99

{

class Program

{

static void Main(string[] args)

{

int[] a = new int[30];

int l = 1;

for (int i = 0; i <=l+1; i++)

{

a[i] = Convert.ToInt16(Console.ReadLine());

l = a[0];

}

int b = UserProgramCode.findExistence(a);

if(b==1)

Console.WriteLine("present");

else if(b==0)

Console.WriteLine("not present");

else

Console.WriteLine("Non Positive");

}

}

class UserProgramCode

{

public static int findExistence(int[] a)

{

int c = 0;

int flag=0;

for (int i = 1; i <= a[0]; i++)

{

if (a[i] >= 0)

{

if (a[i] == a[a[0] + 1])

c++;

}

else

{

flag = 1;

break;

}

}

if (flag == 0)

{

if (c > 0)

return 1;

else

return 0;

}

else

return -1;

}

}

}

**102.GENERATE THE SERIES**

Generate the series  
   Given a method taking an odd positive Integer number as input, write code to evaluate the following series:  
 1+3-5+7-9…+/-n.   
    
 Include a class UserProgramCode with a static method addSeries which accepts a positive integer . The return type of this method is an integer .  
    
 Create a class Program which would get the input as a positive integer and call the static method addSeries present in the UserProgramCode.  
    
 Input and Output Format:  
 Input consists of a positive integer n.  
 Output is a single integer .  
 Refer sample output for formatting specifications.  
    
 Sample Input 1:  
 9  
 Sample Output 1:  
 -3  
 Sample Input 2:  
 11  
 Sample Output 2:  
 8

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace level1\_59**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**int n,x;**

**n = Convert.ToInt32(Console.ReadLine());**

**x = UserProgramCode.addSeries(n);**

**Console.WriteLine(x);**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace level1\_59**

**{**

**class UserProgramCode**

**{**

**public static int addSeries(int a)**

**{**

**int t = 0, k = 1;**

**for (int i = 0; k <= a; i++)**

**{**

**if (i == 0)**

**{**

**t = t + k;**

**}**

**else if (i == 1)**

**{**

**t = t + k;**

**}**

**else if (i % 2 != 0)**

**{**

**t = t + k;**

**}**

**else**

**{**

**t = t - k;**

**}**

**k = k + 2;**

**}**

**return t;**

**}**

**}**

**}**

**103.CONCATENATE STRING**

Concatenate String   Write a program to concatenate two strings as per the following rules. Rules: 1.If the 2 strings are of same length, simply append them together and return the final string. 2.If the 2 given strings are of different length, remove starting characters from the longer string so that both strings are of same length and then append them together and return the final string.  Include a class UserProgramCode with a static method concatstring that accepts a string and returns a string.   
 Create a Class Program which would be used to read 2 strings and call the static method present in UserProgramCode.  
 Input and Output Format:  
 Input consists of two Strings. Output consists of a String.

Sample Input 1:  
 Hello hi

Sample Output 1:

Lohi

Sample Input 2:  
 cognizant coh

Sample Output 2:

Antcoh

Sample Input 3:  
 Hello hello

Sample Output 3:

Hellohello

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace progm54**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**string inp1 = Console.ReadLine();**

**string inp2 = Console.ReadLine();**

**string output = UserProgramCode.concatstring(inp1, inp2);**

**Console.WriteLine(output);**

**}**

**}**

**class UserProgramCode**

**{**

**public static string concatstring(string inputstr1, string inputstr2)**

**{**

**int x = inputstr1.Length;**

**int y = inputstr2.Length;**

**string ans;**

**if (x == y)**

**{**

**ans = inputstr1 + inputstr2;**

**}**

**else if (x > y)**

**{**

**int z = x - y;**

**string inputstring1 = inputstr1.Remove(0, z);**

**ans = inputstring1 + inputstr2;**

**}**

**else**

**{**

**int z = y - x;**

**string inputstring2 = inputstr2.Remove(0, z);**

**ans = inputstr1 + inputstring2;**

**}**

**return ans;**

**}**

**}**

**}**

**104.COUNT THE DIGITS**

Count Digits Write a method to find number of digits present in the given string. Example: Input1: Hell00 ho9 are u Output1: 3 Include a class UserProgramCode with static method countDigits which accepts String value. The return type should be  int.  
 Create a class Program which would get the input and call the static method countDigits present in the UserProgramCode.  
 The input String consists only alphabets , numeric values and whitespaces (blank spaces).Otherwise display as "Invalid Input". Input Output Format: Input consists of a String. Output consists of an integer which counts the number of digits in the given String.

Sample Input 1:

12345

Sample Output 1:

Number of digits present in given string are 5.

Sample Input 2:

hai12!

Sample Output 2:

Invalid Input

countdigits

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace count\_the\_digits

{

class Program

{

static void Main(string[] args)

{

string str = Console.ReadLine();

int op = UserProgramCode.CountDigits(str);

if (op == -1)

Console.WriteLine("Invalid Input");

else

Console.WriteLine("Number of digits present in the given string are {0}.",op);

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace count\_the\_digits

{

class UserProgramCode

{

public static int CountDigits(string str)

{

int c = 0;

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

{

if (!char.IsLetterOrDigit(str[i])&&str[i]!=' ')

{

return -1;

}

if (char.IsDigit(str[i]))

{

c++;

}

}

return c;

}

}

}

**105.CONVERT FORMAT**

Convert Format   Write a program to convert a 10 digit positive number which is in the format XXX-XXX-XXXX  to the format XX-XX-XXX-XXX. Include a class UserProgramCode with a static method convertFormat that accepts a string and returns a string. Create a Class Program which would be used to read the string call the static method present in UserProgramCode. Input and Output Format: Input consists of a String. Output consists of a string

Sample Input 1:

555-555-0000

Sample Output 1:

55-55-550-000

Sample Input 2:

000-000-0000

Sample Output 2:

00-00-000-000

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConvertFormat

{

class Program

{

static void Main(string[] args)

{

string str = Console.ReadLine();

string op = UserProgramCode.ConvertFormat(str);

Console.WriteLine(op);

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConvertFormat

{

class UserProgramCode

{

public static string ConvertFormat(string str)

{

int c = 0;

StringBuilder sb = new StringBuilder();

int len = str.Length;

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

{

if (str[i] == '-')

{

continue;

}

if (Char.IsDigit(str[i]))

{

sb.Append(str[i]);

c++;

}

if (c % 2==0)

{

sb.Append("-");

}

}

sb.Remove((sb.Length) - 1, 1);

string final = sb.ToString();

return final;

}

}

}

**106.ADD & REVERSE**

Given an int array and a number 'k' as input, write a program to add all the elements in the array greater than the given number 'k'. Finally reverse the digits of the obtained sum and print it.  
    
 Include a class UserProgramCode with a static method “addAndReverse” that accepts 2 arguments and returns an integer.The first argument corresponds to the integer array and the second argument corresponds to the number k.  
    
 Create a class Program which would get the required input and call the static method addAndReverse present in the UserProgramCode.  
    
 Example:   
 Input Array = {10,15,20,25,30,100}  
 Number = 15  
 sum = 20 + 25 + 30 + 100 = 175  
 output = 571  
    
 Input and Output Format:  
 The first line of the input consists of an integer that corresponds to the number of elements in the array.  
 The next n lines of the input consists of integers that correspond to the elements in the array.  
 The last line of the input consists of an integer that corresponds to the number, k.  
    
 Output consists of a single integer.  
    
 Sample Input  
 6  
 10  
 15  
 20  
 25  
 30  
 100  
 15  
    
 Sample Output  
 571

using System;

using System.Collections.Generic;

using System.Text;

using System.Linq;

public class Program

{

public static void Main(string[] args)

{

int n = Convert.ToInt32(Console.ReadLine());

int[] ar = new int[n];

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

{

ar[i] = Convert.ToInt32(Console.ReadLine());

}

int k = Convert.ToInt32(Console.ReadLine());

int res = UserMainCode.AddReverse(ar, k);

Console.WriteLine(res);

Console.Read();

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

class UserMainCode

{

public static int AddReverse(int[] ar,int k)

{

int sum = 0,rem,rev=0;

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

{

if (ar[i] > k)

{

sum = sum + ar[i];

}

}

while (sum > 0)

{

rem = sum % 10;

rev = rev \* 10 + rem;

sum = sum / 10;

}

return rev;

}

}

**107.FIXED POINT**

Given an input array of n distinct integers sorted in ascending order, write a program that finds a Fixed Point in the array . Fixed Point in an array is an index i such that arr[i] is equal to i. Business Rules : 1. If any of the given inputs contain any negative number, then print -1. 2. If there are no fixed point values found in the input array, then print -2. 3. If there are less than 2 elements or more than 10 elements in the input array, then print -3.  
    
 Assume that there will be a maximum of 1 fixed point in the input array.    
 Create a class named UserProgramCode that has the following static method   
 public static int findFixedpoint(int[] input1)  
 Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.    
 Input and Output Format:  
 The first line of the input consists of an integer, n that corresponds to the number of elements in the input array 1.  
 The next 'n' lines of input consist of elements in the input array 1.  
 Refer business rules and sample output for formatting specifications.

Sample Input 1 :

6 1 4 45 3 0 19    
 Sample Output1 :  
 3

Sample Input 2 :  
 5 1 10 5 2 -7    
 Sample Output 2 :  
 -1

FIXED POINT

using System;

class Program

{

public static void Main( string[] args )

{

int n = Convert.ToInt32(Console.ReadLine());

int[] array = new int[n];

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

array[i] = Convert.ToInt32(Console.ReadLine());

int result=UserProgramCode.findFixedpoint(array);

Console.WriteLine(result);

Console.Read();

}

}

using System;

class UserProgramCode

{

public static int findFixedpoint(int[] input)

{

Array.Sort(input);

int res=0;

int count=0;

foreach (int k in input)

{

//Console.WriteLine(k);

if (k < 0)

{

count++;

return -1;

}

}

if (input.Length < 2 || input.Length > 10)

{

count++;

return -3;

}

if (count == 0)

{

int i = 1;

foreach(int k in input)

{

if (k==i)

res = i-1;

i++;

}

if (res == 0)

res = -2;

}

return res;

}

}

**108.ONLINE SALES**

An online shopping portal announced a big bang sale,the discounts apply based on purchased time. The discount sales would start from 10 am and will end by 6pm. The discount is not applicable when the products are purchased outside the window time. A) If the product is bought between 10am - 11am,then customer gets 50% off. B) If the product is bought after 11am but within 12pm,then customer gets 40% off. C) If the product is bought after 12pm but within 4pm,then customer gets 30% off. D) If the product is bought after 4pm,only 25% off. The actual price and the time of buying the product are given as input1 and input2 respectively. The time is given as integer in the format as hhmm where hh refers to the hours in 24 hrs time format and mm refers to the minutes . Write a program to calculate the discounted price of the product and print the output in the following format. The actual price of the product is Rs XXX and you have bought it for Rs YYY. You Save Rs ZZZ. Here XXX refers to the actual price of the product, YYY refers to the price after the discount is applied, and ZZZ refers to the difference in between the actual and the discounted price if any. Business rules: 1) If the actual price is zero or less than zero, Print 'Invalid Price Amount' 2) If the product is bought outside the window time , print the output in the following format : The price of the product is Rs XXX and discounts are not applicable.    
 Example 1: input1 : 20000 input2 : 1538(3 PM 38 mins) output : The actual price of the product is Rs 20000 and you have bought it for Rs 14000. You Save Rs 6000. Example 2: input1 :-40 input2 : 1038(10 AM 38 mins) output1 : Invalid Price Amount    
 Create a class named UserProgramCode that has the following static method   
 public static void onlineSalesAmount(int input1,int input2)  
    
 Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.  
    
 Input and Output Format:    
 The first line of the input consists of an integer that corresponds to the cost.  
 The second line of the input consists of an integer that corresponds to the time.  
 Refer business rules and sample output for output format.  
 Display the price after discount as an integer only.  
    
 Sample Input 1 :  
 20000 1538  
    
 Sample Output 1 :

The actual price of the product is Rs 20000 and you have bought it for Rs 14000. You Save Rs 6000.

Sample Input 2 :

-40 1038    
 Sample Output 2 :  
 Invalid Price Amount

using System;

using System.Text.RegularExpressions;

namespace code1

{

class Program

{

static void Main(String[] args)

{

int input1;

int input2;

input1=int.Parse(Console.ReadLine());

input2 = int.Parse(Console.ReadLine());

UserMainCode.onlineSalesAmount(input1, input2);

}

}

}

-------------------------------------------------------------------------------------------

using System;

public class UserMainCode

{

public static void onlineSalesAmount(int input1, int input2)

{

//Console.WriteLine(input1);

if (input1 > 0)

{

if (input2 > 1000 && input2 <= 1100)

{

Console.WriteLine("The actual price of the product is Rs " + input1 + " and you have bought it for Rs " + (input1 - (input1 \* 0.5)) + ". You Save Rs " + (input1 \* 0.5) + ".");

}

else if (input2 > 1100 && input2 <= 1200)

{

Console.WriteLine("The actual price of the product is Rs " + input1 + " and you have bought it for Rs " + (input1 -(input1 \* 0.4)) + ". You Save Rs " + (input1 \* 0.4) + ".");

}

else if (input2 > 1200 && input2 <= 1600)

{

Console.WriteLine("The actual price of the product is Rs " + input1 + " and you have bought it for Rs " + (input1 -(input1 \* 0.3)) + ". You Save Rs " + (input1 \* 0.3) + ".");

}

else if (input2 > 1600 && input2 <= 1800)

{

Console.WriteLine("The actual price of the product is Rs " + input1 + " and you have bought it for Rs " + (input1 -(input1 \* 0.25)) + ". You Save Rs " + (input1 \* 0.25) + ".");

}

else

Console.WriteLine("The price of the product is Rs "+input1+" and discounts are not applicable.");

}

else

Console.WriteLine("Invalid Amount");

}

}

**109.FIND THE SHORTEST STRING**

Write a program that reads an Integer (size of the String List), a String List and a character. Find the shortest string from the list that starts with the character. (case sensitive). Assume there will be only one string.  
 Include a class UserProgramCode with static method  GetshortestString()  that accepts a string list and a character . The return type is String.  
 Create a class Program which would get the input and call the static method GetshortestString(List<string> input1, char input2)) present in the UserProgramCode.  
    
 Input and Output Format :  
 The first input corresponds to the list size.  
 The next input corresponds to the elements in the list which is a string.  
 The third input is a character.  
 Output is String.    
 In GetshortestString()  
 1. Only alphabets should be given in the List else return "Non Alphabets Exists". 2. If there is no match found in input then return "No String Found".  
 3. Otherwise return the appropriate result. In Program Display the result which is return by GetshortestString()

Sample Input 1:

4 read elegant Edit even e

Sample Output 1:

even

Sample Input 2:

2 qwerty abcdef e

Sample Output 2:

No String Found

Sample Input 3:

4 re@d el3gant Edit even e

Sample Output 3:

Non Alphabets Exists

**NO ANSWER FOUND**

**110.SORT STRING**  
    
 Given a string list as input{StringOne, StringTwo, StringThree,...}, Write a program to sort each character in the individual string in ascending order of alphabets {eginnOrSt, ginorStTw, eeghinrrStT,..}, then remove repetitive characters irrespective of case {eginOrSt, ginorStw, eghinrST,..}, then sort the resultant list in ascending order and print the array of string in lowercase{eghinrst,eginorst, ginorstw,..}. Ignore the case sensitivity of given input. Business Rule: 1. If any of the elements in the given input contains any special characters or numbers, print 'Invalid Input'. 2. If there are duplicate elements in the given input,, remove the duplicates and follow all other steps. Example1 : input : 4 Ccaat rat dog cow output : act art cow dgo Steps: 1. Sort each character of each string element: {aacCt,art,dgo,cow} 2. Remove repetitive characters irrespective of the case: {aCt,art,dgo,cow} 3. Now sort each string element and arrange it in ascending order in lowercase: {act,art,cow,dgo} Example 2 : input 4 cat rat dog1 cow$ output Invalid Input Example 3 : input 5 cat rat cat cow rat output act art cow    
 Create a class named UserProgramCode that has the following static method   
 public static List<string> sortString(List<string> arr) Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.    
    
 Input and Output Format:  
 The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.  
 The next 'n' lines of input consist of elements in the input array.  
 Refer business rules and sample output for formatting specifications.

Sample Input :  
 4 Ccaat rat dog cow    
 Sample Output :

act art cow dgo

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace sort\_String

{

class UserprogramCode

{

public static List<string> sortString(List<string> arr)

{

List<StringBuilder> list1 = new List<StringBuilder>();

char temp;

foreach (var st in arr)

{

string str = st.ToLower();

StringBuilder sb = new StringBuilder(str);

for (int i = 0; i < sb.Length - 1; i++)

{

for (int j = i + 1; j < sb.Length; j++)

{

if (sb[i] > sb[j])

{

temp = sb[i];

sb[i] = sb[j];

sb[j] = temp;

}

}

}

list1.Add(sb);

}

foreach (var sb in list1)

{

for (int i = 0; i < sb.Length - 1; i++)

{

if (sb[i].ToString().ToLower() == sb[i + 1].ToString().ToLower())

{

sb.Remove(i, 1);

}

}

}

StringBuilder tempo;

for (int i = 0; i < list1.Count-1; i++)

{

for (int j = i+1; j < list1.Count; j++)

{

if (list1[i][0] > list1[j][0])

{

tempo = list1[i];

list1[i] = list1[j];

list1[j] = tempo;

}

}

}

List<string> final = new List<string>();

foreach (var str in list1)

{

final.Add(str.ToString());

}

return final;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace sort\_String

{

class Program

{

static void Main(string[] args)

{

int n = int.Parse(Console.ReadLine());

List<string> list = new List<string>();

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

{

list.Add(Console.ReadLine());

}

List < string > op= UserprogramCode.sortString(list);

foreach (var item in op)

{

Console.WriteLine(item);

}

}

}

}

**111.CALCULATE CHARGE**

A parking garage charges a Rs.20 minimum fee to park for up to three hours. The garage charges an additional Rs. 5 per hour for each hour or part thereof in excess of three hours. The maximum charge for any given 24-hour period is Rs.100. Assume that no car parks for longer than 24 hours at a time.   
 Write a program which accepts two DateTime inputs as string datatype. Convert the inputs to DateTime DataType and calculate the parking charge for the vehicle.   
    
 Validations:  
 1. DateTime String format is “yyyy-MM-dd:HH:mm:ss” eg: 2009-10-21:14:35:45 .  
 Return -1 as error code for other formats. The first parameter in the method will refer to the checkinDate and the second would refer to checkoutDate.   
 2. CheckoutDateTime should be greater than check in time. Return -2 if that is not the case.  
 3. If the duration exceeds 24 hrs return -3 as error code.   
    
 Include a class UserProgramCode with a static method calculateCharge which accepts two Strings. The return type (Integer) should return the parking charge. Also follow the validations.  
 Create a Class Program which would be used to accept two Strings, and call the static method present in UserProgramCode.  
 Input and Output Format:  
 Input consists of two Strings, the first String corresponds to the CheckinDateTime and the second String corresponds to the CheckoutDateTime.  
 Output consists of an Integer (the parking charges) or, a String “Invalid Date format” if -1 is returned, “CheckoutDateTime is less than CheckinDateTime” if -2 is returned, “Duration exceeds 24 hrs” if -3 is returned.  
    
 Refer sample output for formatting specifications.  
    
 Sample Input 1:  
 2009-10-21:14:35:45  
 2009-10-21:16:35:45  
 Sample Output 1:  
 20  
    
 Sample Input 2:  
 2009-10-211:14:35:45  
 2009-10-21:16:35:45  
 Sample Output 2:  
 Invalid Date format  
    
    
 Sample Input 3:  
 2009-10-21:14:35:45  
 2009-10-21:10:35:45  
 Sample Output 3:  
 CheckoutDateTime is less than CheckinDateTime  
    
    
 Sample Input 4:  
 2009-10-20:14:35:45  
 2009-10-21:16:35:45  
 Sample Output 4:  
 Duration exceeds 24 hrs  
 **using System;**

**class Program**

**{**

**public static void Main( string[] args )**

**{**

**string date1 = Console.ReadLine();**

**string date2 = Console.ReadLine();**

**int result = UserProgramCode.getDateDifference(date1, date2);**

**Console.WriteLine(result);**

**Console.ReadLine();**

**}**

**}**

**using System;**

**using System.Text;**

**using System.Linq;**

**using System.Text.RegularExpressions;**

**class UserProgramCode**

**{**

**public static int getDateDifference(string date1,string date2)**

**{**

**int result=0;**

**int time = 0;**

**DateTime dt1;**

**DateTime dt2;**

**bool res1 = DateTime.TryParseExact(date1, "yyyy-MM-dd:HH:mm:ss", null, System.Globalization.DateTimeStyles.None, out dt1);**

**bool res2 = DateTime.TryParseExact(date2, "yyyy-MM-dd:HH:mm:ss", null, System.Globalization.DateTimeStyles.None, out dt2);**

**if (res1 == true && res2 == true)**

**{**

**time = (int)dt2.Subtract(dt1).TotalHours;**

**if (time < 0)**

**{**

**result= -2;**

**}**

**else if (time > 24)**

**{**

**result= -3;**

**}**

**else**

**{**

**if (time <= 3)**

**{**

**result = 20;**

**}**

**else**

**{**

**result = time \* 5;**

**}**

**if (result > 100)**

**{**

**result = 100;**

**}**

**}**

**}**

**else**

**{**

**result = -1;**

**}**

**return result;**

**}**

**}**

**112.SORT ARRAY ELEMENT**

Sort Array Element Write a method to sort input string array by the length of each element. If word length is same then sort thiese two words in ascending order without considering length.  
 Include a class UserProgramCode with a static method sortArrayElement which accepts a string array as input. The return type of a method is string array. If input contains any special characters then add '-1' into the list.    
 Create a class Program which would get the input and call the static method sortArrayElement present in the UserProgramCode.    
 Input and Output format : The first line of the input consists of an integer that corresponds to the number of elements in the string array. The nexr 'n' lines consist of string inputs. Output consists of array which contains sorted elements or "-1".

Sample Input 1 :

3 Greenapple Litchi Mango

Sample Output 1 : Mango Litchi Greenapple

Sample Input 2 : 2 one #two

Sample Output 2 : -1

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

namespace ConsoleApplication18

{

class Program

{

static void Main(string[] args)

{

int n = Convert.ToInt32(Console.ReadLine());

string[] a = new string[n];

string[] b;

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

a[i] = Console.ReadLine();

b = userProgramCode.sortArrayElement(a);

foreach(string c in b)

Console.WriteLine(c);

Console.ReadLine();

}

}

class userProgramCode

{

public static string[] sortArrayElement(string[] a)

{

string[] b=new string[1];

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

{

for (int j = 0; j < a[i].Length; j++)

{

if (!char.IsLetterOrDigit(a[i][j]))

{

b[0] = "-1";

return b;

}

}

}

Array.Sort(a, StringComparer.Ordinal);

Array.Reverse(a);

return a;

}

}

}

**113. REMOVE TENS**

Given an input array, write a program to remove all Tens present in the array and shift the other elements towards left and fill the trailing empty positions by 0 so that the modified array is of the same length of the given array a  
    
 Include a class UserProgramCode with static method removeTens that accepts an integer array and its size and returns the modified array.  
 Create a class Program which would get the input array and call the static method removeTens present in the UserProgramCode.  
 Input and Output Format:  
 Input consists of an n+1 integers. The 1st integer corresponds to n, the size of the array. The remaining n integers correspond to the element in the array.  
 Output is the modified integer array.  
    
 Sample Input 1:  
 5  
 1 10 20 10 2  
 Sample Output 1:  
 1 20 2 0 0  
    
 Sample Input 2:  
 2 1 2  
    
 Sample Output 2:  
 1 2

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace tens

{

class Program

{

static void Main(string[] args)

{

int n = Convert.ToInt32(Console.ReadLine());

int[] a = new int[n];

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

{

a[i] = Convert.ToInt32(Console.ReadLine());

}

int[] op = new int[100];

op = UserProgramCode.removeTens(a, n);

foreach (int item in op)

Console.WriteLine(item);

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace tens

{

class UserProgramCode

{

public static int[] removeTens(int[] arr, int size)

{

int count = 0, k = 0;

int j;

int[] output = new int[size];

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

{

if (arr[i] != 10)

{

output[k] = arr[i];

k++;

}

else

{

count++;

}

}

for (j = 0; j < count; j++)

{

output[k] = 0;

k++;

}

return output;

}

}

}

**114. SUM OF SQUARES & CUBES OF ELEMENTS IN THE ARRAY**

Write a program to get an int array as input and identify even and odd numbers. If number is odd get cube of it, if number is even get square of it. Finally add all cubes and squares together and return it as output.   
    
 Include a class UserProgramCode with a static method addEvenOdd which accepts an integer array as input and returns an integer.  
 The method returns an integer which is the sum of cubes of all odd numbers and squares of all even numbers in the array.  
    
 Create a class Program which would get the input and call the static method addEvenOdd present in the UserProgramCode.  
    
 Input and Output Format:  
 The first line of the input consists of an integer n, that corresponds to the number of elements in the array. The next 'n' lines of input consists of the elements in the array.  
 Output is an integer that corresponds to the required sum.   
 Refer sample output for formatting specifications.  
    
 Sample Input 1:  
 5  
 2  
 6  
 3  
 4  
 5  
 Sample Output 1:  
 208

63

class Program

{

static void Main(string[] args)

{

int n;

n = Convert.ToInt32(Console.ReadLine());

int[] a=new int[n];

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

{

a[i] = Convert.ToInt32(Console.ReadLine());

}

userprogramcode obj = new userprogramcode();

n=obj.addEvenOdd(a);

Console.WriteLine(n);

}

}

public class userprogramcode

{

public int addEvenOdd(int[] a)

{

int sum=0;

foreach (var n in a)

{

if (n % 2 == 0)

sum += Convert.ToInt32(Math.Pow(n, 2));

else

sum += Convert.ToInt32(Math.Pow(n, 3));

}

return sum;

}

}

115**. BONUS CALCULATION**

Write a program to calculate the bonus of the employee given the basic salary of the employee. The bonus will be calculated based on the below category. If Basic Salary>15000 and less than 20001 calculate bonus as 17% of basic+1500 If Basic Salary>10000 and less than 15001 calculate bonus as 15% of basic+1200 If Basic Salary<10001 calculate bonus as 12% of basic+1000 for rest calculate bonus as 8%of basic+500 Business rule: 1) If the salary given is a negative number, then print -1. 2) If the salary given is more than 1000000, then print -2 . 3) All the test cases has the calculated bonus as integer value only. Create a class named UserProgramCode that has the following static method   
 public static int calculateBonus(int input1)  
 Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.    
 Input and Output Format:  
 Input consists of an integer that corresponds to the salary.  
 Output is an integer.  
 Refer sample output and business rule for output formatting specifications.  
    
 Sample Input 1 :

10000    
 Sample Output1 :  
 2200

Sample Input 2 :  
 2000000  
 Sample Output 2 :  
 -2

calculate bonus

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace bonus

{

class Program

{

static void Main(string[] args)

{

int basic = int.Parse(Console.ReadLine());

int op = UserProgramCode.CalculateBonus(basic);

Console.WriteLine(op);

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace bonus

{

class UserProgramCode

{

public static int CalculateBonus(int basic)

{

int bonus = 0;

if (basic < 0)

return -1;

if (basic > 1000000)

return -2;

if (basic < 20001 && basic > 15000)

{

bonus=Convert.ToInt32(basic\*0.17)+1500;

}

if(basic>10000&&basic<15001)

bonus = Convert.ToInt32(basic \* 0.15) + 1200;

if(basic<10001)

bonus = Convert.ToInt32(basic \* 0.12) + 1000;

else

bonus = Convert.ToInt32(basic \* 0.08) + 500;

return bonus;

}

}

}

116.**INITIAL FORMAT**

Write a program to input a person's name in the format "FirstName LastName" and return the person name in the following format - "LastName, InitialOfFirstName".  
    
 Include a class UserProgramCode with a static method nameFormatter which accepts a string. The return type (string) should return the expected format.  
 Create a Class Program which would be used to accept Input String and call the static method present in UserProgramCode.  
 Input and Output Format:  
 Input consists of a string that corresponds to a Person's name.  
 Output consists of a string(person's name in expected format).  
 Refer sample output for formatting specifications.  
 Sample Input :  
 Jessica Miller    
 Sample Output:  
 Miller, J

**class Program**

**{**

**static void Main(string[] args)**

**{**

**string s;**

**s = Console.ReadLine();**

**userprogramcode obj = new userprogramcode();**

**Console.WriteLine(obj.nameFormatter(s));**

**}**

**}**

**public class userprogramcode**

**{**

**public string nameFormatter(string s)**

**{**

**string[] str;**

**str = s.Split(' ');**

**s = str[1] + ", " + str[0][0];**

**return s;**

**}**

**}**

**117.EXTRACT MAX SUBSTRING**   
 Write method to get a the substring with maximum number of characters for given string input (input1) separated by given delimeter (input2). If two or more substrings have maximum number of characters return the substring which appears first in the alphabetical order. Example : Input1 = "delhi-pune-patna" Input2 = "-" Output1 = "delhi"  
 Create a class named UserProgramCode that has the following static method   
 public static string extractMax(string input1, string input2)  
 Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.    
 Input and Output Format:  
 Input consists of 2 strings.  
 Output is a string.  
    
 Sample Input : delhi-pune-patna  
 -    
 Sample Output :  
 delhi

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace Extract\_max\_SubString1**

**{**

**class UserProgramCode**

**{**

**public static string extractMax(string str, string limit)**

**{**

**List<string> list = new List<string>();**

**string[] arr = str.Split(char.Parse(limit));**

**int max = 0;**

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

**{**

**if (arr[i].Length > max)**

**max = arr[i].Length;**

**}**

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

**{**

**if (arr[i].Length == max)**

**list.Add(arr[i]);**

**}**

**list.Sort();**

**return list[0];**

**}**

**}**

**}**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**namespace Extract\_max\_SubString1**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**string str = Console.ReadLine();**

**string str2 = Console.ReadLine();**

**string op = UserProgramCode.extractMax(str, str2);**

**Console.WriteLine(op);**

**Console.ReadLine();**

**}**

**}**

**}**

**118.FIND LOWEST**

Write a program to find the lowest number in an integer array.  
 Print the lowest number.  
 Only positive number should be given as input in an array. Else print “Negative numbers present”.  
    
 Include a class UserProgramCode with a static method findLowest which accepts an Integer array. The return type (Integer) should return the lowest number. If negative numbers are present in the array, then return -1.  
 Create a Class Program which would be used to accept an Integer and an Integer array, and call the static method present in UserProgramCode.  
 Input and Output Format:  
 Input consists of n+1 Integers, where the first number corresponds the size of the array, followed by the array elements.  
 Output consists of an Integer, the lowest number, or a String “Negative numbers present” if a negative number is present in the array.  
    
 Refer sample output for formatting specifications.  
    
 Sample Input 1:  
 4  
 2  
 3  
 1  
 8  
 Sample Output 1:  
 1  
    
 Sample Input 2:  
 4  
 2  
 3  
 -1  
 8  
 Sample Output 2:  
 Negative numbers present

45. Find Lowest

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Workout45

{

class UserProgramCode

{

public static int findLowest(int[] a)

{

int i, j, t, n;

n = a.Length;

t = a[0];

for (i = 0; i < n; i++)

{

for (j = i + 1; j < n+1; j++)

{

if (a[i] < 0)

{

t = -1;

return t;

}

else if (t > a[i])

{

t = a[i];

a[i] = a[j];

a[j] = t;

}

}

}

return t;

}

}

class Program

{

static void Main(string[] args)

{

//UserProgramCode u = new UserProgramCode();

int i,n,s;

n = int.Parse(Console.ReadLine());

int[] a = new int[n];

for (i = 0; i < n;i++ )

a[i] = int.Parse(Console.ReadLine());

s = UserProgramCode.findLowest(a);

if(s>=0)

Console.WriteLine(s);

else if (s < 0)

Console.WriteLine("Negative Numbers present");

else { }

}

}

}

**119.CHECK SUM**

Write program to read a positive int as input and to calculate the sum of the odd digits in the given number. If the sum is odd print “Odd” else print “Even”.  
    
 Example:   
 input = 56895  
 Sum = 5 + 9 + 5 = 19 (odd)   
 output = Odd  
    
 Include a class UserProgramCode with a static method checkSum which accepts a positive Integer. The return type (Integer) should return 1 if the sum is odd, else return -1.  
 Create a Class Program which would be used to accept a positive Integer, and call the static method present in UserProgramCode.  
 Input and Output Format:  
 Input consists of an Integer.  
 Output consists of a String “Odd” if the sum is odd, else print “Even”.  
    
 Refer sample output for formatting specifications.  
    
 Sample Input:  
 56895  
 Sample Output  
 Odd

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace question36

{

class Program

{

static void Main(string[] args)

{

int n, c;

n = Convert.ToInt32(Console.ReadLine());

c = UserProgramCode.checkSum(n);

if (c == 1)

Console.WriteLine("Odd");

else Console.WriteLine("Even");

}

}

class UserProgramCode

{

public static int checkSum(int a)

{

int rem, sum = 0;

while (a > 0)

{

rem = a % 10;

if (rem % 2 == 1)

{ sum = sum + rem; }

a = a / 10;

}

if (sum % 2 == 1)

return (1);

else return (-1);

}

}

}

**120. CALCULATE DISCOUNT**

Calculate Discount Given two input integer arrays input1 and input2 which contains the details of users booking cars online in the format of key value pairs. input1 (Key,Value) = (UserId1,Bookingmonth1,UserId2,Bookingmonth2,...so on) input2 (Key,Value) = (UserId1,BookingAmount1,UserId2,BookingAmount2,...so on) Write a Program to calculate the discount amount based on the Booking amount and Booking month for every UserId by considering the respective discount rate offered to them and store the result in an output integer array .    
 The discount amount should be calculated using the following specifications ------------------------------------------------------------------------------------------------------------------ Discount Rate                                                           Booking Amount     ------------------------------------------------------------------------------------------------------------------                                                                    |    >=500000        |  >=100000 and < 500000 ------------------------------------------------------------------------------------------------------------------ Discount Rate for Jan-April                     |             25     |            15 Discount Rate for May-August               |            20      |            10 Discount Rate for Sep-December       |  15              |             5 ------------------------------------------------------------------------------------------------------------------ Output should be in the below format: output1 (Key,Value) = (UserId1,DiscountAmount1,UserId2,DiscountAmount2,...so on) Note: 1)Formula: Discount Amount = Booking Amount \* (Discount Rate/100) BUSINESS RULE: 1.If Booking Amount is less than 100000, then discount amount should be zero. 2.If any of the booking month is invalid, then print -1. 3.If any of the booking amount is negative, then print -2. 4.If any of the Booking Month or Booking Amount is missing for any UserId, then print -3 . Create a class named UserProgramCode that has the following static method   
 public static int[] calcDiscount(int[] input1, int[] input2)  
 Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.    
 Input and Output Format:  
 The first line of the input consists of an integer, m that corresponds to the number of elements in the input array 1 .  
 The next 'm' lines of input correspond to elements in the input array 1.  
 The next line of the input consists of an integer, n that corresponds to the number of elements in the input array 2 .  
 The next 'n' lines of input correspond to elements in the input array 2.  
 Refer business rules and sample output for formatting specifications. Sample Input 1 :  
 8 1010 11 1011 02 1012 07 1013 09 8 1010 700000 1011 300000 1012 150000 1013 100000    
 Sample Output 1 : 1010 105000 1011 45000 1012 15000 1013 5000    
 Sample Input 2 : 8 1010 11 1011 02 1012 07 1013 09 6 1010 700000 1011 300000 1012 100000    
 Sample Output 2: -3  

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication7

{

class Program

{

static void Main(string[] args)

{

int m, n,i=0;

m = Convert.ToInt32(Console.ReadLine());

int[] a = new int[m];

int[] output = new int[m];

for (i = 0; i < m; i++)

{

a[i] = Convert.ToInt32(Console.ReadLine());

}

n = Convert.ToInt32(Console.ReadLine());

int[] b = new int[n];

for (i = 0; i < n; i++)

{

b[i] = Convert.ToInt32(Console.ReadLine());

}

output = UserProgramCode.calcDiscount(a, b);

if (output[0] < 0)

{

Console.WriteLine(output[0]);

}

else

{

for (i = 0; i < m; i++)

{

Console.WriteLine(output[i]);

}

}

Console.Read();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication7

{

class UserProgramCode

{

public static int[] calcDiscount(int[] input1, int[] input2)

{

int len1 = 0, len2 = 0,i=0;

int discount=0;

len1 = input1.Length;

len2 = input2.Length;

int[] c = new int[len1];

if (len1 != len2)

{

c[0] = -3;

return c;

}

else

{

for (i = 1; i < len1; i = i + 2)

{

if (!(input1[i] >= 1 && input1[i] <= 12))

{

c[0] = -1;

return c;

}

}

for (i = 1; i < len2; i = i + 2)

{

if ((input2[i] < 0))

{

c[0] = -2;

return c;

}

}

for (i = 1; i < len1; i=i+2)

{

if (input2[i] < 100000)

{

discount = 0;

}

if (input2[i] >= 500000)

{

if (input1[i] >= 1 && input1[i] <= 4)

{

discount = input2[i] \* 25;

}

if (input1[i] >= 5 && input1[i] <= 8)

{

discount = input2[i] \* 20;

}

if (input1[i] >= 9 && input1[i] <= 12)

{

discount = input2[i] \* 15;

}

}

if ((input2[i] >= 100000) && (input2[i]<500000))

{

if (input1[i] >= 1 && input1[i] <= 4)

{

discount = input2[i] \* 15;

}

if (input1[i] >= 5 && input1[i] <= 8)

{

discount = input2[i] \* 10;

}

if (input1[i] >= 9 && input1[i] <= 12)

{

discount = input2[i] \* 5;

}

}

c[i] = discount/100;

}

for (i = 0; i < len1; i=i+2)

{

c[i] = input1[i];

}

return c;

}

}

}

}