

1.

### **BMI Calculator**

Write a program to find the BMI of a person given their height(In Metres) and weight(In Kg) as inputs.

Example :

input1 = 70

input2 = 1.65 Metres

BMI :=  $70 / (1.65 * 1.65) = 25.711$

output = Overweight

Include a class UserProgramCode with static method BMICalc which accepts two float numbers. The return type is String.

Create a class Program which would get the input and call the static method BMICalc present in the UserProgramCode.

Input and Output Format:

Input1 is a float - Weight(In Kg)

Input2 is a float - Height (In Metres)

Output is a string – Interpreted BMI value.

Metric BMI Formula

$$\text{BMI} = ( \text{Weight in Kilograms} / ( \text{Height in Meters} \times \text{Height in Meters} ) )$$

Business rule:

BMI Interpretation is given below

Underweight = BMI of <18.5

Normalweight = BMI of 18.5–24.9

Overweight = BMI of 25–29.9

Obesity = BMI of 30 or greater

If zero or negative number is given as input then return "InvalidInput" , otherwise return "Underweight","Normalweight",

"Overweight", "Obesity" as per Business rule.

Sample Input 1:

70

1.65

Sample Output 1:

Overweight

Sample Input 2:

45

1

Sample Output 2:

Obesity

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace levelI01
```

```
{
```

```
    class Program
```

```
    {
```

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

```
        {
```

```

float input1, input2;

input1 = float.Parse(Console.ReadLine());
input2 = float.Parse(Console.ReadLine());

Console.WriteLine(UserProgramCode.BMISCalc(input1, input2));

}

}

class UserProgramCode
{
    public static string BMISCalc(float input1, float input2)
    {

        float bmi;

        if (input1 <= 0 || input2 <= 0)
        {
            return("InvalidInput");
        }
        else
        {

            bmi = (input1 / (input2 * input2));
            if (bmi < 18.5)
                return("Underweight");
            else if (bmi >= 18.5 && bmi <= 24.9)
                return("Normalweight");
            else if (bmi >= 25 && bmi <= 29.9)
                return("Overweight");
            else if (bmi >= 30)
                return("Obesity");
        }
    }
}

```

```

        }
        return ("null");
    }

}
}

```

## 2. **Time Validation**

Write code to validate time using the following rules:

Business rules:

- It should be a valid time in 12 hrs format
- It should have case insensitive AM or PM
- The time as input in the following format 'hh:mm am' or 'hh:mm pm'

Example:

input = 09:59 pm

output = Valid time format

Include a class UserProgramCode with static method validateTime which accepts the String. The return type should be integer.

Create a class Program which would get the input and call the static method validateTime present in the UserProgramCode.

If the given time is as per the given business rules return 1 else return -1. If the method returns 1 then print "Valid time format" else print "Invalid time format" in Program.

Input and Output Format:

The input time will be a string

Output will be a string. ("Valid time format" or "Invalid time format").

Sample Input 1:

09:59 pm

Sample Output 1:

Valid time format

Q2.Time Validation

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

namespace levelI_02
{
    class Program
    {
        static void Main(string[] args)
        {
            string str = Console.ReadLine();

            int ans = UserProgramCode.validateTime(str);

            if (ans == 1)
                Console.WriteLine("Valid time format");
            else if (ans == -1)
                Console.WriteLine("Invalid time format");
        }
    }

    class UserProgramCode
    {
        public static int validateTime(string str)
        {
```

```

int hr, min;

hr = int.Parse(str.Substring(0, 2));
min = int.Parse(str.Substring(3, 2));
string suf = str.Substring(5, 3);

if (hr > 12 || min > 60 || suf != " am" && suf != " pm")
    return -1;
else
    return 1;

}
}
}

```

### 3. **IP Validator**

Write code to read an IP address in a String variable and validate the IP address. Print “Valid” if it is a valid IP address else print “Invalid”.

Note: An IP address has the format a.b.c.d where a,b,c,d are numbers between 0-255

Include a class UserProgramCode with a static method ipValidator which accepts a string. The return type (integer) should return 1 if it a valid IP, else return 2.

Create a Class Program which would be used to accept a string and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a String.

Output consists of a String(“Valid” or “Invalid”).

Refer sample output for formatting specifications.

Sample Input 1:

132.145.184.210

Sample Output 1:

Valid

Sample Input 2:

132.145.184.290

Sample Output 2:

Invalid

Q3.IP Validator

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;

namespace IP_Validator
{
    class Program
    {
        static void Main(string[] args)
```

```

        {
            Console.WriteLine("IP Address:");
            string ip = Console.ReadLine();
            int res = UserProgramCode.valid(ip);
            if(res==1)
                Console.WriteLine("valid");
            else
                Console.WriteLine("invalid");
            Console.ReadKey();
        }
    }

    class UserProgramCode
    {
        public static int valid(string ip)
        {
            IPAddress address;
            bool res =(IPAddress.TryParse(ip, out address));
            if (res)
                return 1;
            else
                return 0;
        }
    }
}

```

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace level2

```

{
    class Program
    {
        static void Main(string[] args)
        {
            string str = Console.ReadLine();

            int no = UserProgramCode.ipValidator(str);

            if(no==1)

```



```

        Console.WriteLine("Valid");
    else
        Console.WriteLine("Invalid");
    }
}

class UserProgramCode
{

    public static int ipValidator(string str)
    {

        Int32 i = 0, len, no, flag=0;
        len = str.Length;

        Int32 start = 0, count = 0;
        while (i < len)
        {
            if (str.ElementAt(i) != '.')
            {
                count++;
            }
            else
            {
                no = Int32.Parse(str.Substring(start, count));
                if (no < 0 || no > 255)
                    return 2;
                start = start + count+1;

                flag++;
                count = 0;
            }
        }
    }
}

```

```

    }
    i++;
}

no = Int32.Parse(str.Substring(start, count));
flag++;

if (no < 0 || no > 255)
    return 2;
else if (flag > 4)
    return 2;
else
{
    if (i == len)
    {
        return 1;
    }
    else
        return 0;
}}

}

}

```

#### 4. **List the Elements**

Write a program that accepts integer list and an integer. List all the elements in the list that are greater than the value of given integer. Print the result in descending order.

Example:

input1: [1,4,7,3,9,15,24]

input2: 17

Output1:[24]

Include a class UserProgramCode with static method GetElements() which accepts an integer list and the integer as input and returns an integer list.

If there is no element found in input1, then store -1 to the first element of output list.

Create a class Program which would get the input and call the static method GetElements() present in the UserProgramCode. If there is no such element in the input list, print "No element found".

Input and Output Format:

Input consists of n+2 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

The last input is an integer.

Output is an integer list or the string "No element found".

Sample Input 1:

7

1

4

7

3

9

15

24

17

Sample Output 1:

24

Sample Input 2:

6

5

9

3

4

16

21

9

Sample Output 2:

21

16

Q4.List the Elements

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace level104
```

```

{
class Program
{
    static void Main(string[] args)
    {
        int n,i;

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

        int[] arr = new int[n];

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

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

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

        int[] ans = UserProgramCode.GetElements(arr, limit);

        if(ans[0]== -1)

            Console.WriteLine("No element found");

        else

        {

            foreach (int item in ans)

            {

                Console.WriteLine(item);

            }

        }

    }

}

class UserProgramCode
{

    public static int[] GetElements(int[] arr,int limit)

    {

        int n = arr.Length;

        int[] temp=new int[n];

```

```

    int i,j;
    i=0;
    for(j=0;j<n;j++)
    {
        if(arr[j]>limit)
        {
            temp[i]=arr[j];
            i++;
        }
    }

    if (temp[0] == 0)
    {
        temp[0] = -1;
        return temp;
    }

    else
    {
        Console.WriteLine("");
        Array.Sort(temp);
        Array.Reverse(temp);
        Array.Resize(ref temp, i);
        return temp;
    }
}

```

Write a program to calculate the division/class obtained by the student when the marks obtained by a student in 5 different subjects are given as inputs.

The student gets a division/class as per the following rules:

Percentage above or equal to 60 - "First Class".

Percentage between 50 and 59 - "Second Class".

Percentage between 40 and 49 - "Third Class".

Percentage less than 40 - "Failed".

Include a class `UserProgramCode` with a static method `calculateResult` which accepts five integers. The return type (`String`) should return the class of the student.

Create a Class Program which would be used to accept 5 integer inputs and call the static method present in `UserProgramCode`.

Input and Output Format:

Input consists of five integers.

Output consists of a `String`(class of the student).

Refer sample output for formatting specifications.

Sample Input 1:

41

45

46

40

41

Sample Output 1:

Third Class

## Q5.Class Division

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

namespace level105
{
    class Program
    {
        static void Main(string[] args)
        {
            int sub1, sub2, sub3, sub4, sub5;
            sub1 = int.Parse(Console.ReadLine());
            sub2 = int.Parse(Console.ReadLine());
            sub3 = int.Parse(Console.ReadLine());
            sub4 = int.Parse(Console.ReadLine());
            sub5 = int.Parse(Console.ReadLine());

            Console.WriteLine(UserProgramCode.calculateResult(sub1, sub2, sub3, sub4, sub5));
        }
    }
    class UserProgramCode
    {
        public static string calculateResult(int sub1,int sub2,int sub3,int sub4,int sub5)
        {
            // int sub1,sub2,sub4,sub3,sub5;
```



```

int sum=sub1+sub2+sub3+sub4+sub5;

string str="";
int avg=sum/5;
if(avg>=60)
    str="First Class";
else if(avg>=50 && avg<=59)
    str="Second Class";
else if(avg>=40 && avg<=49)
    str="Third Class";
else if(avg<40)
    str="Failed";
return str;

}

}

}

```

## 6. **Index power array**

Write code to read an integer array and to find the power of each individual element according to its position index, add them up and print as output.

Example :

input = {7,6,2,1}

output = (7 power 0)+(6 power 1)+(2 power 2)+(1 power 3) = 1+6+4+1=12

Include a class `UserProgramCode` with a static method `getSumOfPower` which accepts an integer that corresponds to the size of the array and an integer array. The return type (`Integer`) should return the final output.

Create a Class Program which would be used to accept Input array 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 number of elements, followed by the array elements.

Output consists of an `Integer`(final output).

Refer sample output for formatting specifications.

Sample Input 1:

4

7

6

2

1

Sample Output 1:

12

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Reflection;
```

```
namespace ConsoleApplication2
```

```

{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            if (n > 0)
            {
                int[] a = new int[n];
                for(int i=0;i<n;i++)
                {
                    a[i] = int.Parse(Console.ReadLine());
                }
                int sum = UserProgramCode.getSumOfPower(n, a);
                Console.WriteLine(sum);
            }
        }
    }
}

```

```

class UserProgramCode
{
    public static int getSumOfPower(int size,int[] a)
    {
        int sum=0;
        for(int i=0;i<size; i++)
        {

```

```

        sum+=(int)Math.Pow(a[i],i);

    }
    return sum;
}
}

```

### 7.Count of Elements

Write a program that gets the count of elements in input1 list that starts with the character passed in input2 irrespective of case. Print the count.

Example:

input1: ['abc','Apple','Mango']

input2: a

Output1:

2

Business Rule:

1. If there is no element that start with the given char in input1, then return -1.
2. Only alphabets should be given in input1 string else return -2.

Include a class UserProgramCode with a static method GetCount which accepts the size of the string array, string array and a character. The return type (Integer) should return count. Follow the Business rules.

Create a Class Program which would be used to accept the size of the array, the array elements and a character, 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, a string list, and a character.

Output consists of an Integer(final count), or a String(“No elements Found” if -1 is returned or “Only alphabets should be given” if -2 is returned.

Refer sample output for formatting specifications.

Sample Input 1:

3

abc

Apple

Mango

a

Sample Output 1:

2

Sample Input 2:

2

goods

bad\$

a

Sample Output 2:

No elements Found

Sample Input 3:

2

good\$

bad\$

a

Sample Output 3:

Only alphabets should be given

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Reflection;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            string[] str = new string[n];
            if (n > 0)
            {
                for (int i = 0; i < n; i++)
                {
                    str[i] = Console.ReadLine();
                }
                char c = char.Parse(Console.ReadLine());
                int output = UserProgramCode.getCount(n, str, c);
                if (output > 0)
                {
                    Console.WriteLine(output);
                }
                else if (output == -1)
                {
                    Console.WriteLine("No elements Found");
                }
            }
        }
    }
}
```

```

    }
    else if (output == -2)
    {
        Console.WriteLine("Only alphabets should be given");
    }
}

}

}

}

```

```

class UserProgramCode
{
    public static int getCount(int size,string[] str,char c)
    {
        int count=0;
        char c_cap = c;
        char c_small = c;
        Regex reg = new Regex(@"^([A-Za-z]{1,})$");
        foreach (string s in str)
        {

            if (!reg.IsMatch(s))
                return -2;
            string ch = c.ToString();
            if (c >= 97 && c <= 122)
            {
                c_cap = (char)((int)(c) - 32);
            }
        }
    }
}

```

```

    }
    else if (c >= 65 && c <= 90)
    {
        c_small = (char)((int)(c) + 32);
    }

    char[] inp = s.ToCharArray();

    if (c_small == inp[0] || c_cap == inp[0])
        count++;

    }
    if (count >= 1)
    {
        return count;
    }
    else
        return -1;
    }
}

```

```

-----

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;

namespace ConsoleApplication32
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());

```



```

        string[] s = new string[n];
        for (int i = 0; i < n; i++)
        {
            s[i] = Console.ReadLine();
        }
        string c = Console.ReadLine();
        int f = user.res(n, s, c);
        if(f== -1)
            Console.WriteLine("no elements");
        else if(f== -2)
            Console.WriteLine("only give alphabets");
        else
            Console.WriteLine(f);
    }
}

class user
{
    public static int res(int n, string[] s, string c)
    {
        int count = 0;
        Regex r = new Regex(@"^[A-Za-z]*$");
        for (int i = 0; i < s.Length; i++)
        {
            if (!r.IsMatch(s[i]))
            {
                return -2;
            }
        }

        for (int i = 0; i < s.Length; i++)
        {
            s[i].ToLower();
        }
        for (int i = 0; i < s.Length; i++)
        {
            if (s[i].StartsWith(c, StringComparison.OrdinalIgnoreCase))
            {
                count++;
            }
        }
        if(count==0)
        {
            return -1;
        }
        return count;
    }
}
}

```

---

Write a program to read a String and to replace every appearance of the word "is" by "is not". If the word "is" is immediately preceded or followed by a letter no change should be made to the string. Print the final string.

Example:

input = This is just a misconception

output = This is not just a misconception

Include a class UserProgramCode with a static method negativeString which accepts a string. The return type (String) should return the final output.

Create a Class Program which would be used to accept a string input, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

Sample input 1:

This is just a misconception

Sample Output 1:

This is not just a misconception

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
using System.Reflection;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            string s = Console.ReadLine();
            string output = UserProgramCode.negativeString(s);
            Console.WriteLine(output);

        }
    }
}

class UserProgramCode
{
    public static string negativeString(string str)
    {
        string neg_string = null;
        string[] str1 = str.Split(' ');
        StringBuilder sb = new StringBuilder();
        for (int i=0;i<str1.Length;i++)
        {

            if (str1[i].Equals("is"))
            {
```

```

        sb.Append("is not ");
    }
    else
    {
        sb.Append(str1[i]+" ");
    }
}
neg_string = sb.ToString();
return neg_string;
}
}

```

9)

### 9.Sum of Squares

Write a program to find the sum of the squares of first n natural numbers. If n less than 0, return -1.

Include a class UserProgramCode with a static method sumSquare which accepts an integer. The return type is an integer as given in the above statement.

Create a Class Program which would be used to accept Input and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of the value n.

Output consists of a integer as mentioned in the problem statement.

Refer sample output for formatting specifications.

Sample Input 1:

3

Sample Output 1:

14

Sample Input 2:

-5

Sample Output 2:

-1

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Reflection;
```

```
namespace ConsoleApplication2
```

```
{  
    class Program  
    {
```

```

static void Main(string[] args)
{
    int n = int.Parse(Console.ReadLine());
    int result=UserProgramCode.sumSquare(n);
    if ( result== -1)
    {
        Console.WriteLine(-1);
    }
    else
    {
        Console.WriteLine(result);
    }

}
}
}

```

```

class UserProgramCode
{
    public static int sumSquare(int n)
    {
        int sum = 0;
        if (n < 0)
            return -1;
        for (int i = 1; i <= n; i++)
        {
            sum += (int)Math.Pow(i, 2);
        }
    }
}

```

```
        return sum;
    }
}
```

10)

### 10.Cattle Graze

In a village there is a ground with full of grass where the cattle-rearing people take their cattle to maze in the ground. Assume that the cattle is tied to a tree. Write a program to calculate the area of grass that the cattle can maze. The rope length would be the input and area rounded of two decimal places would be the output.

Do not use Math.PI for the value of PI. Use 3.14 directly.

Include a class UserProgramCode with a static method calculateArea which accepts an integer. The return type is double. The method returns the area rounded to 2 decimal places.

Create a Class Program which would be used to accept Input and call the static method present in UserProgramCode.

Use random function in Math library.

Input and Output Format:

Input consists of the integer value n.

Output consists of a double.

Refer sample output for formatting specifications.

Sample Input 1:

3

Sample Output 1:

28.26

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Reflection;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            Console.WriteLine(UserProgramCode.calculateArea(n).ToString("#0.00"));

        }
    }
}
```



```
class UserProgramCode
{
    public static double calculateArea(int n)
    {
        double area = 0;
        area = Math.Round((3.14*n*n),2);
        return area;
    }
}
```

## 11. **Reverse Substring**

Given a input string with a startIndex and length, Write a program to extract substring from right to left. Assume the last character has index 0.

Include a class UserProgramCode with a static method reverseSubstring which accepts a string and two integers. The return type is string as given in the above statement.

Create a Class Program which would be used to accept Input and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a string, and two integers – startIndex and length.

Output consists of a string as mentioned in the problem statement.

Refer sample output for formatting specifications.

Sample Input 1:

rajasthan

2

3

Sample Output 1:

hts

```
using System;

    using System.Collections.Generic;
    using System.Linq;
    using System.Text;

namespace ConsoleApplication9
{
    public class UserProgramCode
    {
        public static string reverseSubstring(string str, int start, int len)
        {
            StringBuilder sb = new StringBuilder();

            char[] ch = str.ToCharArray();
            Array.Reverse(ch);
            foreach (char item in ch)
            {
                sb.Append(item);
            }
            string s1 = sb.ToString();
            string s2 = s1.Substring(start, len);
            return s2;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            string str = Console.ReadLine();
```

```

        int start = int.Parse(Console.ReadLine());
        int len = int.Parse(Console.ReadLine());
        string str2 = UserProgramCode.reverseSubstring(str, start, len);
        Console.WriteLine(str2);
    }
}

```

## 12. Shipping Cost

Write a program to compute the Cost of Booking for Shipping. The Shipping Cost is computed according to the shipping type and the package weight. The shipping rate is given below.

Shipping types - Weight Rate (bahts/gram)

Regular for first 2000 - 0.25 (basic charge)

Regular exceeding 2000 - 0.35

For each Express, use the same rate as Regular + 50 bahts fee

Note that the Shipping cost is computed from the possible valid minimum rate.

Input1- Weight in grams

Input2- Type of delivery ('R' Regular and 'X' Express)

Example:

Input1: 4500

Input2: R

Output1: 1375

Include a class UserProgramCode with a static method CalcShippingCost which accepts an integer(weight) and a character (type of delivery). The return type (integer) should return the shipping cost.

Create a Class Program which would be used to accept a integer value and a character, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of an integer and a character.

Output consists of an integer(the shipping cost).

Refer sample output for formatting specifications.

Sample input 1:

4500

R

Sample Output 1:

1375

Sample Input 2:

1800

X

Sample Output 2:

500

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

namespace rate
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            char ch = char.Parse(Console.ReadLine());
            Console.WriteLine(UserCode.CalcShippingCost(n,ch));
        }
    }
}

class UserCode
{
    public static double CalcShippingCost(int gms, char ch)
    {
        double charge;
        if (ch == 'R' && gms <= 2000)
```

```

    {
        charge = (gms * .25);
    }
    else if (ch == 'R' && gms >= 2000)
    {
        charge = (gms - 2000) * .35 + (2000 * .25);
    }
    else if (ch == 'E' && gms <= 2000)
    {
        charge = (gms * .25) + 50;
    }
    else if (ch == 'E' && gms >= 2000)
    {
        charge = (gms - 2000) * .35 + (2000 * .25) + 50;
    }
    else
    {
        return 0;
    }

    return charge;
}
}

```

---

### 13. Valid Negative Number

Write a program to read a negative number as a String variable and to validate the number. If the given string contains a valid negative number print corresponding positive number else print "Invalid number".

Example:

input = "-94923"

output = "94923"

Include a class UserProgramCode with a static method validateNumber which accepts a String. The return type (String) should return the corresponding output. If the input string is not a valid negative number, the method returns "-1".

Create a Class Program which would be used to accept a String, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a String( a negative number).

Output consists of a String(the corresponding output).

Refer sample output for formatting specifications.

Sample Input 1:

-94923

Sample Output 1:

94923

Sample Input 2:

-13O

Sample Output 2:

Invalid number

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication9

{

public class UserProgramCode

{

public static string validateNumber(string str)

{

str.ToCharArray();

int temp = 0;

if (str[0] == '-')

```

    {
        for (int i = 0; i < str.Length; i++)
            if (str[i] >= 48 && str[i] <= 57)
                temp = 1;
            else
                temp = 0;
        if (temp == 1)
        {
            str = str.Substring(1, str.Length-1);
            return str.ToString();
        }
        else
            return "-1";
    }
    else
        return "-1";
}
}

class Program
{
    static void Main(string[] args)
    {
        string str = Console.ReadLine();
        Console.WriteLine(UserProgramCode.validateNumber(str));
    }
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;

```

```

namespace rate
{
    class Program
    {
        static void Main(string[] args)
        {
            string str = Console.ReadLine();
            Console.WriteLine(UserCode.validateNumber(str));
        }
    }
}

class UserCode
{
    public static string validateNumber(string str)
    {
        Regex r = new Regex(@"^[0-9]+");
        if (!r.IsMatch(str))
            return "Invalid";
        else
        {
            string s1 = str.Substring(1, str.Length - 1);
            return s1;
        }
    }
}

```

-----

#### 14. Add non Common Elements

Write a program to read two integer arrays and to add all the non common elements from the 2 integer arrays. Print the final output.

Example:

input1: [7,9,1,0]

input2: [10,6,5]

Output1:38

Business Rules:

Only positive numbers should be given to the input Lists.

1. If the input1 List consists of negative numbers, return -1.



2. If the input2 List consists of negative numbers, return -2.
3. If the both the input lists consists of negative numbers, return -3.

Include a class `UserProgramCode` with a static method which accepts the inputs in the following order (input1, size1, input2, size2). The return type (integer) should return output according to the business rules.

Create a Class Program which would be used to accept two lists, and call the static method present in `UserProgramCode`.

Input and Output Format:

Input consists of  $n+m+2$  integers, where first two integers corresponds to the size of the two array lists, respectively, followed by the corresponding array elements.

Output consists of an Integer(the corresponding output), or a String "Input 1 has negative numbers" if the first array contains negative numbers, "Input 2 has negative numbers" if the second array contains negative numbers, or "Both inputs has negative numbers" if both array has negative numbers.

Refer sample output for formatting specifications.

Sample Input 1:

4  
3  
6  
9  
2  
1  
10  
7  
5

Sample Output 1:

38

Sample Input 2:

4  
3  
-6  
9  
2  
1  
10  
7  
5

Sample Output 2:

Input 1 has negative numbers

Sample Input 3:

4  
3  
6  
9  
2  
1  
10  
-7  
5

Sample Output 3:

Input 2 has negative numbers

Sample Input 3:

4  
3  
6

9

-2

1

10

-7

5

Sample Output 3:

Both inputs has negative numbers

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

namespace non_common_
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            int m = int.Parse(Console.ReadLine());
            int[] a1 = new int[n];
            int[] a2 = new int[m];
            for (int i = 0; i < n; i++)
            {
                a1[i] = int.Parse(Console.ReadLine());
            }
            for (int i = 0; i < m; i++)
            {
                a2[i] = int.Parse(Console.ReadLine());
            }

            int flag = UserCode.sumNonCommonElement(a1, n, a2, m);
            if(flag== -1)
                Console.WriteLine("Input 1 has negative numbers");
            else if(flag==-2)
                Console.WriteLine("Input 2 has negative numbers");
            else if(flag==-3)
                Console.WriteLine("Both inputs has negative numbers");
            else
                Console.WriteLine(flag);
        }
    }
}
```

```

    }
}

class UserCode
{
    public static int sumNonCommonElement(int[] a1, int n, int[] a2, int m)
    {
        int sum=0,a=0,b=0;
        for (int i = 0; i < n; i++)
            if (a1[i] < 0)
                a = 1;
            else
                a=0;
        for (int j = 0; j < m; j++)
            if (a2[j] < 0)
                b=1;
            else
                b=0;

        if (a == 0 && b == 0)
        {
            int[] op = a1.Except(a2).Union(a2.Except(a1)).ToArray();
            foreach (int item in op)
            {
                sum = sum + item;
            }
            return sum;
        }
        if (a == 1 && b == 0)
            return -1;
        else if (b == 1 && a == 0)
            return -2;
        if (a == 1 && b == 1)
            return -3;

        return 0;
    }
}

```

### 15.Get the longest string

Write a program to get the longest string from the list which starts with the given character. Assume that input comparison is done irrespective of case. ie case insensitive.

Include a class UserProgramCode with a static method getLongestString which accepts a String list and a character. The return type is a string.

Create a Class Program which would be used to accept the size of the string list, the list elements and the search character and calls the static method present in UserProgramCode.

In getLongestString

1. If there is no element found list, then return the string "No elements found "
2. Only alphabets should be given in the list. Otherwise return the string, "String contains non alphabetic characters. "
3. If the two or more strings start with the given character ,the longest string should be returned. Assume that the longest string will be unique.

Input Output format

First line points to the size of the string list as n.

The next n lines points to elements of the string list.

The last input points to the character.

Output consists of a string.

SAMPLE INPUT 1:

4

Yellow

Red

Black

Blue

b

SAMPLE OUTPUT 1:

Black

SAMPLE INPUT 2:

3

Black

White

45

W

SAMPLE OUTPUT 2:

String contains non alphabetic characters.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;

namespace mock_pgm2
{
    public class Isletter
    {

        public bool IsAlphaNumeric(string input)
        {
            return Regex.IsMatch(input, "^[a-zA-Z]+$");
        }

    }

    class Program
    {
        static void Main(string[] args)
        {
            Isletter obj=new Isletter();
```

```

int count = 0, temp = 0 , alpha=1;
List<string> ls=new List<string>();
int n=int.Parse(Console.ReadLine());
for(int i=0;i<n;i++)
{
    ls.Add(Console.ReadLine());
    if (!obj.IsAlphaNumeric(ls[i]))
    {
        alpha = 0;
    }
}

char ch = char.Parse(Console.ReadLine());
if (alpha == 0)
{
    Console.WriteLine("String contains non alphabetic characters.");
}
else
{
    for (int i = 0; i < n; i++)
    {
        if (ls[i].ToCharArray()[0] == char.ToUpper(ch) || ls[i].ToCharArray()[0] ==
char.ToLower(ch))
        {
            if (ls[i].Length > count)
                count = ls[i].Length;
            temp = 1;
        }
    }
    if (temp == 0)

```





```

static void Main(string[] args)
{
    int n = int.Parse(Console.ReadLine());
    List<string> list = new List<string>(n);
    Int i;
    for (i = 0; i < n; i++)
    list.Add(Console.ReadLine());
    char c = char.Parse(Console.ReadLine());
    Console.WriteLine(GetlongestString(list, c));
}

```

```

static string GetlongestString(List<string> list, char c)
{
    string ch = c.ToString();
    for (int i = 0; i < list.Count; i++)
    {
        char[] c1 = list[i].ToCharArray();
        foreach (char c2 in c1)
        {
            if (!char.IsLetter(c2))
            {
                return "Non Alphabets Exists";
            }
            else
            {
                continue;
            }
        }
    }
}

```

```
var q = from s in list
where s.StartsWith(ch)
orderby s.Length //descending-longest
select s;
```

```
foreach (var item in q)
{
return item;
}
return "No Elements Found";
}
}
}
```

#### **16. Length of the longest string**

Write code to find the length of the longest string in the given string list.

Include a class UserProgramCode with static method longestWordLength that accepts the String list and the return type should be int

Create a class Program which would get the input and call the static method longestWordLength(String[] array) present in the UserProgramCode.

The longestWordLength(String[] array) returns the length of the longest string

Input and Output Format:

The first integer corresponds to n, the number of elements in the list. The next 'n' integers correspond to the elements in the String list.

SAMPLE INPUT 1

2

Black

Blue

SAMPLE OUTPUT 1

5

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

namespace Fwd_Prgs
{
    public class UserProgramCode
    {
        public static string longestWordLength(string[] s)
        {
            int sum = 0;
            for (int i = 0; i < s.Length; i++)
            {
                if (s[i].Length > sum)
                {
                    sum = s[i].Length;
                }
            }
            return sum.ToString();
        }
    }
}
```

```

class Program
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());
        string[] str=new string[n];
        for (int i = 0; i < n; i++)
        {
            str[i] = Console.ReadLine();
        }
        string res = UserProgramCode.longestWordLength(str);
        Console.WriteLine(res);
    }
}

```

17.

### Get All Elements

Write a program to get all the elements that are greater than 5 from a given input integer list.

Display it in the order as present in the array.

Print the elements.

Example:

Input1: [1,3,7,8,5,13]

Output1:[7,8,13]

Business Rule:

If any of the element in the input list is greater than 500 then store -1 in the oth index of the output list.

Include a class UserProgramCode with a static method GetAllElements which accepts an integer List and its size. The return type (integer list) should return output according to the business rules

Create a Class Program which would be used to accept a list, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of n+1 integers, where first integer corresponds to the size of the list, followed by the corresponding list elements.

Output consists of an Integer list, or a String “Array element greater than 500” if any of the elements is greater than 500.

Refer sample output for formatting specifications.

Sample Input 1:

6

1

3

7

8

5

13

Sample Output1:

7

8

13

Sample Input 2:

6

1

3

7

8

501

13

Sample Output 2:

Array element greater than 500

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace Fwd_Prgs
```

```
{
```

```
    public class UserProgramCode
```

```
    {
```

```
        public static int GetAllElements(List<int> a, int n)
```

```
        {
```

```
            List<int> b=new List<int>();
```

```
            int j = 0;
```

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

```
            {
```

```

        if (a[i] > 500)
            b[0] = -1;
        else if (a[i] > 5)
        {
            b.Add(a[i]);
        }
    }
    b.Sort();
    if (b[0] == -1)
        Console.WriteLine("Array element greater than 500");

    else
        for(int i=0;i<b.Count;i++)
            Console.WriteLine(b[i]);
    return 0;
}

}

class Program
{
    static void Main(string[] args)
    {
        int n= int.Parse(Console.ReadLine());
        List<int> a = new List<int>();
        for(int i=0;i<n;i++)
            a.Add(int.Parse(Console.ReadLine()));
        UserProgramCode.GetAllElements(a,n);
    }
}
}

```

### 18) 18.Sum Common Elements

Write a program to read two int arrays, eg. A{2,3,5,1} and B{1,3,9}, and to find out sum of common elements in given arrays. Print the sum, or print “No common elements found” if there are no common elements.

Assume the common element appears only once in each array.

Include a class UserProgramCode with a static method getSumOfIntersection which accept the size of two integer arrays and the two integer arrays. The return type (integer) should return the sum, or -1, accordingly.

Create a Class Program which would be used to accept two integer arrays, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of n+m+2 integers, where first two integers corresponds to the size of the two array lists, respectively, followed by the corresponding array elements.

Output consists of an Integer(the corresponding output) or string - (“No common elements found”).

Refer sample output for formatting specifications.

Sample Input 1:

4  
3  
2  
3  
5  
1  
1  
3  
9



Sample Output 1:

4

Sample Input 2:

4

3

2

31

5

14

1

3

9

Sample Output 2:

No common elements found

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace Fwd_Prgs
```

```
{
```

```
    public class UserProgramCode
```

```
    {
```

```
        public static int getSumOfIntersection(int n1, int n2, int[] a, int[] b)
```

```
        {
```

```
            int sum=0;
```

```
            for (int i = 0; i < n1; i++)
```

```

{
    for (int j = 0; j < n2; j++)
        if(a[i]==b[j])
            sum = sum + a[i];

}

if (sum == 0)
    return -1;
else
    return sum;
}
}

```

class Program

```

{
    static void Main(string[] args)
    {
        int n1 = int.Parse(Console.ReadLine());
        int n2 = int.Parse(Console.ReadLine());
        int[] a=new int[n1];
        int[] b=new int[n2];
        for(int i = 0; i < n1; i++)
            a[i] = int.Parse(Console.ReadLine());
        for(int i = 0; i < n2; i++)
            b[i] = int.Parse(Console.ReadLine());
        int res = UserProgramCode.getSumOfIntersection(n1, n2, a, b);
        if(res== -1)
            Console.WriteLine("No common elements found");
        else
            Console.WriteLine(res);
    }
}

```

```
    }  
  }  
}
```

19) 19.Find largest digit in a given number

Write a code to find the Largest digit from given input integer.

Include a class UserProgramCode with static method findLargestDigit(int num)

Create a class Program which would get the input and call the static method findLargestDigit(num) present in the UserProgramCode.

If the interger is a negative value findLargestDigit(num) method returns -1 to Program, otherwise returns largest digit in a given number.

If -1 is returned then print "The number should be a positive number".

Input and Output Format:

Input is an integer n.

Output is an integer which is the largest digit in the given number n.

SAMPLE INPUT 1:

456

SAMPLE OUTPUT1:

6

SAMPLE INPUT 2:

-12434567

SAMPLE OUTPUT2:

The number should be a positive number

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

namespace Fwd_Prgs
{
    public class UserProgramCode
    {
        public static int findLargestDigit(int num)
        {
            if (num > 0)
            {
                int temp, res = 0;

                while (num > 0)
                {
                    temp = num % 10;
                    if (temp > res)
                        res = temp;
                    num = num / 10;
                }
                return res;
            }
            else
                return -1;
        }
    }

    class Program
```

```

{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());
        int res = UserProgramCode.findLargestDigit(n);
        if(res!=-1)
            Console.WriteLine(res);
        else
            Console.WriteLine("The number should be a positive number");

    }
}

```

## 20) 20.Sum Of Digits

Write a program to read a String and to get the sum of all the digits present in the given string. Print the sum. If there is no digit in the given string print “No digit present”.

Example1:

Input = good23bad4

Output = 2 + 3 + 4 = 9

Include a class UserProgramCode with a static method sumOfDigits which accepts a String. The return type (Integer) should return the sum, or return -1 if no digits are present.

Create a Class Program which would be used to accept a String and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a string.

Output consists of an Integer or a String "No digit present " ..

Refer sample output for formatting specifications.

Sample Input 1:

good23bad4

Sample Output 1:

9

Sample Input 2:

good@#bad\$

Sample Output 2:

No digit present

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace Fwd_Prgs
```

```
{
```

```
    public class UserProgramCode
```

```
    {
```

```
        public static string sumOfDigits(string str)
```

```
        {
```

```
            char[] s = str.ToCharArray();
```

```
            int sum=0;
```

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

```

        if ((s[i] > 47) && (s[i] < 58))
        {
            sum = sum + (int)s[i]-48;

        }
        string res = sum.ToString();
        if (sum == 0)
            return "-1";
        else
            return res;
    }
}

```

```

class Program
{
    static void Main(string[] args)
    {
        string s = Console.ReadLine();
        string res = UserProgramCode.sumOfDigits(s);
        if(res=="-1")
            Console.WriteLine("No digit present");
        else

            Console.WriteLine(res);
    }
}

```

```
public static void user(string st)
{
    int sum=0;
    char[] c=st.ToCharArray();
    foreach (var item in c)
    {
        if (char.IsDigit(item))
        {
            string s=item.ToString();
            int a=Convert.ToInt16(s);
            sum = sum + a;
        }
    }
    Console.WriteLine(sum);
}
```

---



n..21

## 21.String Encryption

Write code to encrypt the given string using following rules and print the encrypted string:

Rules:

Replace the characters at odd positions by next character in alphabet.

Leave the characters at even positions unchanged.

If an odd position character is 'z' replace it by 'a'.

Assume the first character in the string is at position 1.

Include a class UserProgramCode with static method encrypt that accepts a string and returns the encrypted string.

Create a class Program which would get the input and call the static method encrypt present in the UserProgramCode.

Input and Output Format :

The input is a String .

The output is a String which holds the encrypted text.

Sample Input 1:

curiosity

Sample Output 1:

dusipsjtz

```
class Program
```

```
{
```

```

static void Main(string[] args)
{
    string str = Console.ReadLine();
    string s1=UserProgramCode.method1(str);
    Console.WriteLine(s1);
    Console.ReadLine();
}
}
class UserProgramCode
{
    public static string method1(string s)
    {
        int i = 0, a = 0;
        StringBuilder sb = new StringBuilder();
        for (i = 0; i < s.Length; i++)
        {
            if (i % 2 == 0)
            {
                if (s[i] == 'z')
                {
                    sb.Append('a');
                }
                else
                {
                    a = Convert.ToInt16(s[i]);
                    Console.WriteLine(a);
                    a = a + 1;
                    sb.Append(Convert.ToChar(a));
                }
            }
        }
        else
        {
            sb.Append(s[i]);
        }
    }
}

```

```

    }
    return sb.ToString();
}
}

```

qn..22

## 22. Arithmetic Operation

Write a program to perform the user specified arithmetic operation. The program will consist of 3 parameters. First one for specifying the type of operation (+, -, \*, /) and the other two are the operands upon which the operation has to be performed. Print the final output.

Business Rules:

1. The first parameter should have the values as 1,2,3 or 4. If it has any other value other than this, return -1.
2. The Second and the third parameter should be only positive numbers, else return -2.
3. If the first parameter is

1 Add the second and third parameter. (second+third)

2 Subtract the second and third parameter. (second-third)

3 ----- Multiply second and third parameter. (second\*third)

4 Divide second by third parameter. (second/third)

Include a class `UserProgramCode` with a static method `arithmeticOperation` which accepts three integers. The return type (`Integer`) should return the result of the operation performed. Return -1 or -2 according to the Business rules.

Create a Class Program which would be used to accept three integers, and call the static method present in `UserProgramCode`.

Input and Output Format:

Input consists of three integers, where the first integer corresponds to the type of operator, the second and third integer corresponds to the operands.

Output consists of an Integer or, a String “Invalid operator” if the -1 is returned, “Invalid operands” if -2 is returned.

Assume all outputs are Integers.

Refer sample output for formatting specifications.

Sample Input 1:

1  
2  
3

Sample Output 1:

5

Sample Input 2:

5  
2  
3

Sample Output 2:

Invalid operator

Sample Input 3:

1

-2

3

Sample Output 3:

Invalid operands

```
class Program
```

```
{
    static void Main(string[] args)
    {
        int a = int.Parse(Console.ReadLine());
        int b = int.Parse(Console.ReadLine());
        int c = int.Parse(Console.ReadLine());
        int s1 = UserProgramCode.arithmeticOperation(a,b,c);
        if (s1 == -1)
            Console.WriteLine("Invalid Operator");
        else if (s1 == -2)
            Console.WriteLine("Invalid Operands");
        else
            Console.WriteLine(s1);
        Console.ReadLine();
    }
}
```

```
class UserProgramCode
```

```
{
    public static int arithmeticOperation(int a,int b,int c)
    {
        if (a > 0 && a < 5)
        {
```

```

        if (b < 0 || c < 0)
            return -2;
        else
            if (a == 1)
                return b + c;
            else if (a == 2)
                return b - c;
            else if (a == 3)
                return b * c;
            else
                return b / c;
    }
    else
        return -1;
}
}

```

qn..23

### 23.Get Middle Characters

Write a program to read a string of even length and to fetch two middle most characters. Print the output.

Example:

Input = this

Output1 = hi

Include a class UserProgramCode with a static method getMiddleChars which accepts a String. The return type (String) should return the output String.

Create a Class Program which would be used to accept a String, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a String of even length.

Output consists of a String, the middle two letters

Refer sample output for formatting specifications.

Sample Input 1:

This

Sample Output 1:

hi

```
class Program
{
    static void Main(string[] args)
    {
        string str=Console.ReadLine();

        string s1 = UserProgramCode.getMiddleChars(str);
        Console.WriteLine(s1);

        Console.ReadLine();
    }
}
class UserProgramCode
{
    public static string getMiddleChars(string s)
    {
        int i = 0;
```

```

        StringBuilder sb = new StringBuilder();
        if (s.Length % 2 == 0)
        {
            i = s.Length / 2;
            sb.Append(s[i - 1]);
            sb.Append(s[i]);
        }
        return sb.ToString();
    }
}

```

qn..24

#### 24.Add Days

Write a program which can print the date n days after the given date.

The date should be given in string format “mm/dd/yyyy” without time and the resultant added date should also be in the format “mm/dd/yyyy”.

Only positive value should be given as input to the days to be added, else print “n value is negative”. If the date format is not “mm/dd/yyyy” , then print “Invalid date format” .

Example : 5 days after “10/21/2009” is “10/26/2009”.

Include a class UserProgramCode with a static method addDays which accepts a String and an Integer. The return type (String) should return the final date as String or it would return "-1" if the day value is negative or it would return "-2" if the date is not as per the given format.

Create a Class Program which would be used to accept a String, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a String and an integer, where the String corresponds to the input date and the integer corresponds to the number of days.



Output consists of a String.

Refer sample output for formatting specifications.

Sample Input 1:

10/21/2009

5

Sample Output 1:

10/26/2009

Sample Input 2:

10/21/2009

-5

Sample Output 2:

n value is negative

Sample Input 3:

40/21/2009

5

Sample Output 3:

Invalid date format

```
class Program
```

```
{
```

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

```
    {
```

```
        string s=Console.ReadLine();
```

```
        int i=Convert.ToInt16(Console.ReadLine());
```

```

        string s1 = UserProgramCode.addDays(s,i);
        if (s1 == "-1")
            Console.WriteLine("n value is negative");
        else if (s1 == "-2")
            Console.WriteLine("Invalid date format");
        Console.WriteLine(s1);

        Console.ReadLine();
    }
}
class UserProgramCode
{
    public static string addDays(string s,int a)
    {
        string format = "MM/dd/yyyy";
        DateTime dt;
        bool b=DateTime.TryParseExact(s,format,null,System.Globalization.DateTimeStyles.None,out
dt);
        if (!b)
            return "-2";
        if (a < 0)
            return "-1";
        dt=dt.AddDays(a);

        return dt.ToString("MM/dd/yyyy");
    }
}

```

## 25.Sum Of Squares Of Even Digits

Write a program to read a positive integer and to calculate the sum of squares of even digits available in the given number. Print the output.

Example:

input = 56895

output =  $6*6 + 8*8 = 100$

Include a class UserProgramCode with a static method sumOfSquaresOfEvenDigits which accepts an Integer. The return type (Integer) should return the sum of squares of even digits available in the given number.

Create a Class Program which would be used to accept an Integer, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of an Integer.

Output consists of an Integer, the sum of squares of even digits available in the given number .

Refer sample output for formatting specifications.

Sample Input 1:

56895

Sample Output 1:

100

```
class Program
```

```
{
```

```

static void Main(string[] args)
{

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

    int s1 = UserProgramCode.sumOfSquareofEvenDigits(i);

    Console.WriteLine(s1);

    Console.ReadLine();
}
}
class UserProgramCode
{
    public static int sumOfSquareofEvenDigits(int a)
    {
        int sum=0;
        string a1 = a.ToString();
        int a2 = 0;
        for (int i = 0; i < a1.Length; i++)
        {
            a2 = (int)(a1[i])-48;
            if (a2 % 2 == 0)
            {

                sum = sum + (a2 * a2);

            }
        }
        return sum;
    }
}

```

```
}
```

another method

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace sum_of_squares_of_evndigits
```

```
{
```

```
    class Userprogramcode
```

```
    {
```

```
        public static int sumOfSquareofEvenDigits(int a)
```

```
        {
```

```
            int temp;
```

```
            int res=0;
```

```
            int sum = 0;
```

```
            temp = a;
```

```
            while (temp > 0)
```

```
            {
```

```
                a = temp % 10;
```

```
                if (temp % 2 == 0)
```

```
                {
```

```
                    res = a*a;
```

```
                    sum = sum +res;
```

```
                }
```

```
                temp=temp/10;
```

```
            }
```

```
return sum;
```

```
}
```

```
}
```

```
}
```

-----

qn..26

26.Colour Code

Write a program to find whether the given string corresponds to a valid colour code or not.

Write code to validate the given color code based on following rules:

- Must start with # symbol
- Must contain six characters after #
- It may contain alphabets from A-F (only upper case) or digits from 0-9

Example :

input = #FF9922

output = Valid

Include a class UserProgramCode with a static method validateColorCode. This method returns 1 if the input corresponds to a valid color code. Else this method returns -1.

Create a class Program which would get the input and call the static method validateColorCode present in the UserProgramCode.

Input and Output Format:

Input is a string - color code as value

Output is a string - Valid or Invalid

Sample Input 1:

#FF9922

Sample Output 1:

Valid

Sample Input 2:

1234567

Sample Output 2:

Invalid

```
class Program
{
    static void Main(string[] args)
    {
        string str = Console.ReadLine();
        int i = UserProgramCode.validateColorCode(str);
        if (i == 1)
            Console.WriteLine("Valid");
        else
            Console.WriteLine("Invalid");
    }
}
```

```

        Console.ReadLine();
    }

}

class UserProgramCode
{
    public static int validateColorCode(string s)
    {
        int flag = 0;
        if(s.StartsWith("#"))
        {
            if (s.Length == 7)
            {
                char[] ch = s.ToCharArray();
                for(int i=1;i<=6;i++)
                {
                    if (char.IsDigit(ch[i]) || "ABCDEF".Contains(ch[i]))
                    {
                        flag = 1;
                    }
                    else
                    {
                        flag = 0;
                        break;
                    }
                }
            }
        }
        if (flag == 0)
            return -1;
    }
}

```



```
        else
            return 1;
    }
}
```

another method:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
```

```
namespace colour_code
{
    class Userprogramcode
    {
        public static int validateColorCode(string s)
        {
            Regex reg = new Regex(@"^(([#])+([A-F0-9]{6}))$");
            if(reg.IsMatch(s))
            {
                return 1;
            }
            else
            {
                return -1;
            }
        }
    }
}
```

---

qn..27

## 27.Fibonacci Series

Write method to generate fibonacci series and calculate the sum of first n numbers in the series and return it as output.

Example:

input = 5

output =  $0 + 1 + 1 + 2 + 3 = 7$

Include a class UserProgramCode with a static method getSumOfNfibos that accepts an integer as input and returns an integer.

Create a class Program which would get the input and call the static method getSumOfNfibos present in the UserProgramCode.

Input and Output Format:

Input consists of an integer that corresponds to n.

Output consists of an integer which corresponds to the sum of the first n terms in the fibonacci series.

Note: First two numbers in a Fibonacci series are 0, 1 and all other subsequent numbers are sum of its previous two numbers. Example - 0, 1, 1, 2, 3, 5...

Sample Input 1:

15

Sample Output 1:

986

Sample Input 2:

4

Sample Output 2:

4

```
class Program
{
    static void Main(string[] args)
    {
        int n = Convert.ToInt16(Console.ReadLine());
        int i = UserProgramCode.getSumOfNfibos(n);
        Console.WriteLine(i);
        Console.ReadLine();
    }
}

class UserProgramCode
{
    public static int getSumOfNfibos(int n)
    {
        int f = 0, f1 = -1, f2 = 1, sum = 0;
        for (int i = 0; i < n; i++)
        {
            f = f1 + f2;
            f1 = f2;
            f2 = f;
            sum = sum + f;
        }
    }
}
```

```
    }  
    return sum;  
}  
}
```

qn..28

## 28. Shortest Length

Write a method to get the length of the shortest word in the given string array.

Include a class `UserProgramCode` with a static method `shortestWordLength` that accepts a string array and returns an integer that corresponds to the length of the shortest word.

Create a class `Program` which would get the input and call the static method `shortestWordLength` present in the `UserProgramCode`.

Input and Output Format:

First line of the input consists of an integer that corresponds to the number of elements in the string array.

The next  $n$  lines of the input consists of the elements in the string array. Assume that all the elements in the string array are single words.

Output is an integer which corresponds to the length of the shortest word

Sample Input 1:

```
3  
cherry  
hai  
apple
```

Sample Output 1:

3

Sample Input 2:

4

cherry

apple

blueberry

grapes

Sample Output 2:

5

class Program

```
{  
    static void Main(string[] args)  
    {  
        int i = 0;  
        int n = int.Parse(Console.ReadLine());  
        string[] s = new string[50];  
        for (i = 0; i < n; i++)  
            s[i] = Console.ReadLine();  
        s[i] = "\\0";  
        int sl = UserProgramCode.shortestWordLength(s);  
        Console.WriteLine(sl);  
        Console.ReadLine();  
    }  
}
```

class UserProgramCode

```

{
    public static int shortestWordLength(string[] s)
    {

        int sl=s[0].Length;
        for (int i = 1; s[i]!="\0"; i++)
        {
            if (s[i].Length < sl)
                sl = s[i].Length;

        }
        return sl;
    }
}

```

qn..29

29. Calculate Bill

Write a program to calculate the bill given the previous reading , current reading and per unit charge as inputs.

Example:

input1 =ABC2012345

input2 = ABC2012660

input3 = 4

$$\text{Bill} = (12660 - 12345) * 4$$

output = 1260

Include a class UserProgramCode with static method calculateBill() that accepts 2 strings corresponding to the previous reading and current reading and an integer that corresponds to the per unit charge. This method returns an integer that corresponds to the bill amount to be paid.

Create a class Program which would get the inputs and call the static method calculateBill() present in the UserProgramCode.

Input and Output Format:

Reading Format - XXXXXAAAAA where XXXXX is consumer number and AAAAA is meter reading.

Input1 is a String - previous reading of the consumer

Input2 is a String - current reading of the consumer

Input3 is an integer - per unit charge to the consumer

output is an integer - Calculated BILL value.

Metric BILL Formula:

$$\text{Bill} = (\text{current reading} - \text{previous reading}) * \text{per unit charge}$$

Sample Input 1:

ABC2012345

ABC2012660

4

Sample Output 1:

1260

```
class Program
```

```
{
```

```

static void Main(string[] args)
{
    string s1 = Console.ReadLine();
    string s2 = Console.ReadLine();
    int n = int.Parse(Console.ReadLine());
    int sl = UserProgramCode.calculateBill(s1,s2,n);
    Console.WriteLine(sl);
    Console.ReadLine();
}
}
class UserProgramCode
{
    public static int calculateBill(string s1,string s2,int n)
    {
        int sum = 0;
        string ss1 = s1.Substring(5);
        string ss2 = s2.Substring(5);
        int a = int.Parse(ss1);
        int b = int.Parse(ss2);
        sum = (b - a) * n;
        return sum;
    }
}

```

qn..30

30. Calculate the sum of cube



Write a program to find the sum of the cube of first 'n' natural numbers.

Example:

input = 5

output = 225

Include a class `UserProgramCode` with a static method `sumOfCube()` that accepts an integer and returns an integer . If the input is not a natural number, return -1.

Create a class `Program` which would get the input and call the static method `sumOfCube()` present in the `UserProgramCode`.

Input and Output Format:

Input is an integer that corresponds to n

Output is an integer (Sum of cubes) or if the given input n is not a natural number then print “The input is not a natural number”

Sample Input 1:

5

Sample Output 1:

225

Sample Input 2:

-1

Sample Output 2:

The input is not a natural number

```

class Program
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());
        int s1 = UserProgramCode.sumOfCube(n);
        if(s1==-1)
            console.writeline("The input is not a natural number");
        else
            Console.WriteLine(s1);
        Console.ReadLine();
    }
}

class UserProgramCode
{
    public static int sumOfCube(int n)
    {
        if(n<0)
            return -1;
        int sum = 0;
        for (int i = 1; i <= n; i++)
            sum = sum + (i * i * i);
        return sum;
    }
}

```

//31.form new word

### 31. Form New Word

Write a program to read a string and a positive int (say n) as input and to construct a string with first n and last n characters in the given string. Note - the given string length is  $\geq 2n$

Example:

Input1 = California

input2 = 3

output = Calnia

Include a class UserProgramCode with a static method formNewWord() that accepts a string and an integer. The method returns a string.

Create a class Program which would get the inputs and call the static method formNewWord() present in the UserProgramCode.

Input and Output Format:

Input consists of a string and an integer.

Output is a String that corresponds to the newly formed word.

Sample Input 1:

California

3

Sample Output 1:

Calnia

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

namespace Workout41
{
    class UserProgramCode
    {
        public static string formNewWord(string s,int n)
        {
            string s1,s2;
            int l,n1;
            l = s.Length;
            if (l > n * 2)
            {
                n1 = l - n;
                s1 = s.Substring(0, n) + s.Substring(n1, n);
                return s1;
            }
            else
            {
                return "";
            }
        }
    }

    class Program
```

```

{
    static void Main(string[] args)
    {
        string s;
        int n;
        UserProgramCode u = new UserProgramCode();
        s = Console.ReadLine();
        n = int.Parse(Console.ReadLine());
        s = UserProgramCode.formNewWord(s,n);
        Console.WriteLine(s);

    }
}

```

=====

32. check characters

32.CheckCharacters

Given a method with a string input, write code to test if first and last characters are same. Incase they are same return 1 else return -1 as output. Note - Consider case.

Example:

Input = ""the picture was great""

first character = 't'

last character = 't'

Output = 1

Include a class `UserProgramCode` with static method `checkCharacters` that accepts a string and returns an integer.

Create a class `Program` which would get the input and call the static method `checkCharacters` present in the `UserProgramCode`.

Input and Output Format:

Input is a String - a sentence

Output is a String --- "The characters are same" or "The characters are different".

Sample Input 1:

the picture was great

Sample Output 1:

The characters are same

Sample Input 2:

Hai how are you?

Sample Output 2:

The characters are different

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace program32
```

```
{
```

```

class Program
{
    static void Main(string[] args)
    {
        int n;

        n=UserProgramCode.checkcharacters("the picture was great");

        Console.WriteLine(n);

    }
}

class UserProgramCode
{
    public static int checkcharacters(string str)
    {
        int len = str.Length;
        string str1 = str.Substring(0, 1);
        string str2 = str.Substring(len-1);
        if (str1.Equals(str2))
        {
            return 1;
        }
        else
        {
            return -1;
        }
    }
}
}

```

---

33.count characters

33.Count Characters

Write a program to count the number of characters present in the given input String.

Include a class UserProgramCode with static method countCharacters which accepts string array.

The return type is a integer value.

Create a class Program which would get the input and call the static method countCharacters present in theUserProgramCode .

Sample Input 1:

qwerty

Sample Output 1:

6

Sample Input 2:

12345

Sample Output 2:

5

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```



```
namespace program33
```

```
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int n;  
            n= UserProgramCode.countcharacters("qwerty");  
            Console.WriteLine(n);  
        }  
    }  
    class UserProgramCode  
    {  
        public static int countcharacters(string str)  
        {  
            int len = str.Length;  
            return len;  
        }  
    }  
}
```

=====

34. validate id location

34. Validate ID Locations

Write a program to read two string inputs and check whether the first string is in valid format. First string is ID and second string is location. A valid ID should be in the format CTS-LLL-XXXX where LLL is the first three letters of given location and XXXX is a four digit number. If the given ID is as per the given format, print “valid” else print “invalid”.

Example:

Input1 = CTS-hyd-1234

Input2 = hyderabad

output = valid

Include a class UserProgramCode with a static method validateIDLocations which accepts two Strings. The return value (Integer) should be 1 if the first string is valid, else return -1.

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 2 strings.

Output consists of a string, “valid” or “invalid”.

Refer sample output for formatting specifications.

Sample Input 1:

CTS-hyd-1234

hyderabad

Sample Output 1:

valid

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Text.RegularExpressions;
```

```

namespace prgrm35
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;

            i = UserProgramCode.validateIDlocations("CTS-HYD-2000", "HYDERABAD");

            if (i == 1)
            {
                Console.WriteLine("valid");
            }
            else
            {
                Console.WriteLine("Invalid");
            }

        }
    }
}

class UserProgramCode
{
    public static int validateIDlocations(string s1, string s2)
    {
        int output = 0;

        Regex reg = new Regex(@"^([CTS]+[-]+([A-Za-z]{3})+[-]+([0-9]{4}))$");

        if(reg.IsMatch(s1))
        {
            string res=s2.ToUpper();

            if (s1.Contains(res.Substring(0, 3)))

```

```

        {
            output = 1;
        }
    else
    {
        output = -1;
    }
}
else
{
    output = -2;
}
return output;

}

}

}

```

---

35.string manipulation

35.String Manipulation

Write a program which can read two strings and add the reverse of the second string in the middle of the first string.

Print "Special character found" if the string consists of special characters, else print the final String.

Examples :

1)

String1 : Arun

String2: Ram

Output : ArmaRun

2)

String1 : Aruns

String2: Ram

Output : ArumaRns

Hint: If the first string contains odd number of characters

e.g. String1 is having 7 characters, then add the second reverse string after the 4 characters of the first string.

Include a class UserProgramCode with a static method stringManipulation which accepts two Strings. The return type (String) should return the final String.

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.

Output consists of a String( the final String), or “Special character found” if the string consists of special characters.

Refer sample output for formatting specifications.

Sample Input 1:

Arun

Ram

Sample Output 1:

ArmaRun

Sample Input 2:

Aruns

Ram

Sample Output 2:

ArumaRns

Sample Input 3:

Arun\$

Ram

Sample Output 3:

Special character found

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace prgm34
```

```
{
```

```
    class Program
```

```
    {
```

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

```
        {
```

```

        string ans;

        ans= UserProgramCode.stringmanipulation("sathish", "kumar");

        Console.WriteLine(ans);
    }
}

class UserProgramCode
{
    public static string stringmanipulation(string str1, string str2)
    {
        string res1, res2, res3, rev = "";
        int len, len2, len3;

        len = str1.Length;
        if (len % 2 == 0)
        {
            len2 = len / 2;
            res3=str1.Substring(len2);
        }
        else
        {
            len2 = (len / 2) + 1;
            res3 = str1.Substring(len2);
        }

        len3 = str2.Length-1;
        while (len3 >= 0)
        {
            rev = rev + str2[len3];
            len3--;
        }

        res1 = str1.Substring(0, len2);
        res2 = res1 + rev + res3;
    }
}

```

```
return res2;
```

```
}
```

```
}
```

```
}
```

another method

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace string_manipulation
```

```
{
```

```
class Userprogramcode
```

```
{
```

```
public static string stringmanipulation(string str1, string str2)
```

```
{
```

```
string a;
```

```
char[] c=str2.ToCharArray();
```

```
Array.Reverse(c);
```

```
string d = new string(c);
```

```
if (str1.Length % 2 == 0)
```

```
{
```



```

        a = str1.Substring(0, str1.Length / 2) + d + str1.Substring(str1.Length / 2);

    }
    else
    {
        a = str1.Substring(0, (str1.Length / 2)+1) + d + str1.Substring((str1.Length / 2)+1);

    }
    return a;

}

}

}

```

=====

question 36

### 36. 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);

    }

}

```

}

=====

question 37 :

### 37. Find Gift Voucher

In a game two dice is thrown. From the sum of the two dice, the player is going to get the gift voucher from the club. Write a program to find the amount of the gift voucher. Print the amount received as gift.

Sum of Two Dices ----- Gift Voucher in Rs

2,3,6,11 ----- 1000

4,7,10----- 3000

5,8,9,12----- 5000

In the method,

Only Positive number (1-6) should be given as a input numbers. Else return -1.

Include a class UserProgramCode with a static method findGiftVoucher which accepts two integers. The return type (Integer) should return the gift voucher amount. If the any of the inputs is invalid 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 two integers.

Output consists of an Integer( the gift voucher amount) or a String “Invalid Input” if any of the inputs is invalid.

Refer sample output for formatting specifications.

Sample Input 1:

1

2

Sample Output 1:

1000

Sample Input 2:

1

-2

Sample Output 2:

Invalid Input

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace question36
```

```
{
```

```
    class Program
```

```
    {
```

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

```
        {
```

```
            int n, c,m;
```

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

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

```
c = UserProgramCode.findGiftVoucher(n,m);  
Console.WriteLine(c);
```

```
    }  
}
```

```
class UserProgramCode
```

```
{
```

```
    public static int findGiftVoucher(int a,int b)
```

```
    {
```

```
        if(a>0 && b>0 && a<7 && b<7)
```

```
        {
```

```
            if ((a + b == 2) || (a + b == 3) || (a + b == 6) || (a + b == 11))
```

```
                return (1000);
```

```
            else if ((a + b == 4) || (a + b == 7) || (a + b == 10))
```

```
                return (3000);
```

```
            else if ((a + b == 5) || (a + b == 8) || (a + b == 9) || (a + b == 12))
```

```
                return (5000);
```

```
        }
```

```
        else return(-1);
```

```
        return 0;
```

```
    }
```

```
}
```

```
}
```

```
=====
```

question 38 :

### 38. 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

using System;

using System.Collections.Generic;

using System.Linq;



```
using System.Text;

namespace q38
{
    class Program
    {
        static void Main(string[] args)
        {

            int n=int.Parse(Console.ReadLine());
            List<string> list=new List<string>(n);
            int i;
            for (i = 0; i < n; i++)
                list.Add(Console.ReadLine());
            char c = char.Parse(Console.ReadLine());
            Console.WriteLine(GetshortestString(list, c));

        }
        static string GetshortestString(List<string> list, char c)
        {
            string min = "";
            int len = 99;
            int i;
            for (i = 0; i < list.Count; i++)
            {

                if (list[i][0].CompareTo(c) == 0)
                {

                    if (list[i].Length < len)
```

```

        {
            min = list[i];
            len = list[i].Length;
        }
    }
}

if (len == 99)
    return("No string Found");
else
    return(min);
}
}
}

```

another method:

```

static string GetlongestString(List<string> list, char c)
{
    string ch = c.ToString();
    for (inti = 0; i<list.Count; i++)
    {
        char[] c1 = list[i].ToCharArray();
        foreach (char c2 in c1)
        {
            if (!char.IsLetter(c2))
            {
                return "Non Alphabets Exists";
            }
        }
    }
}

```

```

else
{
continue;
}
}
}

var q = from s in list
where s.StartsWith(ch)
orderby s.Length //descending-longest
select s;
foreach (var item in q)
{
return item;
}
return "No Elements Found";
}
}
}

```

=====

question 39 :

### 39.Reverse Number

Write a program to read a positive number as input and to get the reverse of the given number and print it as output.

Example:

input = 543

output = 345

Include a class UserProgramCode with a static method reverseNumber which accepts an Integer. The return type (Integer) should return the reverse of the given input.

Create a Class Program which would be used to accept an Integer, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of an Integer.

Output consists of an Integer, the reverse of the given Input.

Refer sample output for formatting specifications.

Sample Input 1:

543

Sample Output 1:

345

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace question36
```

```
{
```

```
    class Program
```

```
    {
```

```
static void Main(string[] args)
{
    int n, c=0;
    n = Convert.ToInt32(Console.ReadLine());
    if (n > 0)
    {
        c = UserProgramCode.reverseNumber(n);
    }
    Console.WriteLine(c);
}
}
```

```
class UserProgramCode
{
    public static int reverseNumber(int a)
    {
        int rem, sum = 0;

        while (a > 0)
        {

            rem = a % 10;

            sum = (sum*10) + rem;
            a = a / 10;
        }

        return (sum);
    }
}
```

```
}  
  
}  
  
}
```

=====

program : 40

#### 40.String Finder

Write a program to read three strings which are Searchstring, Str1 and Str2 as input and to find out if Str2 comes after Str1 in the Searchstring. If yes print “Yes” else print “No”.

Example:

input1 = geniousRajKumarDev

input2 = Raj

input3 = Dev

output = Yes

Include a class UserProgramCode with a static method stringFinder which accepts 3 strings. The return type (Integer) should return 1 if the Str2 comes after Str1 in the Searchstring, else return 2.

Create a Class Program which would be used to read 3 strings, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of three strings which are Searchstring, Str1 and Str2.

Output consists of a String, “Yes” or “No”.

Refer sample output for formatting specifications.

Sample Input 1:

geniousRajKumarDev

Raj

Dev

Sample Output 1:

Yes

Sample Input 2:

geniousRajKumarDev

Dev

Raj

Sample Output 2:

No

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace prog40
```

```
{
```

```
    class Program
```

```
    {
```

```

static void Main(string[] args)
{
    string str, str1, str2;
    str = Console.ReadLine();
    str1=Console.ReadLine();
    str2 = Console.ReadLine();
    if(stringFinder(str,str1,str2)==1)
        Console.WriteLine("Yes");
    else
        Console.WriteLine("No");

}

static int stringFinder(string str,string str1,string str2)
{
    int str1_len = str1.Length;
    int str2_len = str2.Length;
    string temp="";
    int st=0,count1=0,count2=0;
    while (temp != str1)
    {
        temp = str.Substring(st, str1_len);
        st++;
    }
    if (temp == str1)
    {
        count1++;
    }
    // Console.WriteLine(temp + "\t" + st);
    string sub = str.Substring((st + str1_len - 1));

```



```

        temp = "";
        st = 0;
        while (temp != str2)
        {
            temp = str.Substring(st, str2_len);
            st++;
        }
        if (temp == str2)
        {
            count2++;
        }
        if(count1==1 && count2==1)
            return 1;
        else
            return 2;
        // Console.WriteLine(temp+"\t"+st);
    }

}

```

another method:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```

namespace string_finder

```

```

{

```

```

class Userprogramcode
{
    public static int stringFinder(string str, string str1, string str2)
    {
        int a = str.IndexOf(str1);
        int b = str.IndexOf(str2);
        if (a < b)
        {
            return 1;
        }
        else
            return -1;
    }
}

```

=====

41. Form New Word:

41. Form New Word

Write a program to read a string and a positive int (say n) as input and to construct a string with first n and last n characters in the given string. Note - the given string length is  $\geq 2n$

Example:

Input1 = California

input2 = 3

output = Calnia

Include a class UserProgramCode with a static method formNewWord() that accepts a string and an integer. The method returns a string.

Create a class Program which would get the inputs and call the static method formNewWord() present in the UserProgramCode.

Input and Output Format:

Input consists of a string and an integer.

Output is a String that corresponds to the newly formed word.

Sample Input 1:

California

3

Sample Output 1:

Calnia

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace Workout41
```

```
{
```

```
    class UserProgramCode
```

```
    {
```

```

public static string formNewWord(string s,int n)
{
    string s1,s2;
    int l,n1;
    l = s.Length;
    if (l > n * 2)
    {
        n1 = l - n;
        s1 = s.Substring(0, n) + s.Substring(n1, n);
        return s1;
    }
    else
    {
        return "";
    }
}
}

```

```

class Program
{
    static void Main(string[] args)
    {
        string s;
        int n;

        UserProgramCode u = new UserProgramCode();
        s = Console.ReadLine();
        n = int.Parse(Console.ReadLine());
        s = UserProgramCode.formNewWord(s,n);
        Console.WriteLine(s);
    }
}

```

```
}  
}  
}
```

#### 42. Check Characters

#### 42. CheckCharacters

Given a method with a string input, write code to test if first and last characters are same. In case they are same return 1 else return -1 as output. Note - Consider case.

Example:

Input = "the picture was great"

first character = 't'

last character = 't'

Output = 1

Include a class UserProgramCode with static method checkCharacters that accepts a string and returns an integer.

Create a class Program which would get the input and call the static method checkCharacters present in the UserProgramCode.

Input and Output Format:

Input is a String - a sentence

Output is a String --- "The characters are same" or "The characters are different".

Sample Input 1:

the picture was great

Sample Output 1:

The characters are same

Sample Input 2:

Hai how are you?

Sample Output 2:

The characters are different

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace Workout42
```

```
{
```

```
    class UserProgramCode
```

```
    {
```

```
        public static int checkCharacters(string s)
```

```
        {
```

```
            string sl;
```

```
            sl=s.Substring(0,1);
```

```
            sl = sl.ToLower();
```

```
            if (s.EndsWith(sl))
```

```
                return 1;
```

```
            else
```

```

        return 0;
    }
}

class Program
{
    static void Main(string[] args)
    {
        string s;
        int i;
        //UserProgramCode u = new UserProgramCode();
        s = Console.ReadLine();
        s = s.ToLower();
        i = UserProgramCode.checkCharacters(s);

        if (i==1)
            Console.WriteLine("The characters are same");
        else
            Console.WriteLine("The characters are different");

    }
}

```

---

43. CountCharacters

43.Count Characters

Write a program to count the number of characters present in the given input String.

Include a class UserProgramCode with static method countCharacters which accepts string array.

The return type is a integer value.

Create a class Program which would get the input and call the static method countCharacters present in theUserProgramCode .

Sample Input 1:

qwerty

Sample Output 1:

6

Sample Input 2:

12345

Sample Output 2:

5

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace Workout43
```

```
{
```

```
    class UserProgramCode
```

```
    {
```

```
        public static int countCharacters(string s)
```

```
        {
```

```
            int l;
```



```

        l = s.Length;
        return l;
    }
}

class Program
{
    static void Main(string[] args)
    {
        //UserProgramCode u = new UserProgramCode();
        string s;
        int n;
        s = Console.ReadLine();
        n = UserProgramCode.countCharacters(s);
        Console.WriteLine(n);

    }
}

```

=====

44. Find total number of days in given month

44.Find Total number of days in given month

Write code to find out total number of days in the given month for the given year.

Month is coded as: Jan=0, Feb=1 ,Mar=2 ...

Include a class UserProgramCode with static method getNumberOfDays that accepts two integers and return type should be int.

Create a class Program which would get the input and call the static method getNumberOfDays(int year, int month) present in the UserProgramCode.

Return the result from getNumberOfDays and display the result in Program class.

Input and Output Format :

The first integer represent the year.

The second integer represents the month

The output is an interger which is number of days in the given month.

SAMPLE INPUT 1:

2000

1

SAMPLE OUTPUT 1:

29

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace Workout44
```

```
{
```

```
    class UserProgramCode
```

```
    {
```

```
        public static int getNumberOfDays(int y,int m)
```

```
        {
```

```
            int d;
```

```
            d = System.DateTime.DaysInMonth(y,m+1);
```

```
            return d;
```

```
        }
```

```

    }

class Program
{
    static void Main(string[] args)
    {
        //UserProgramCode u = new UserProgramCode();

        int y,m,n;

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

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

        n = UserProgramCode.getNumberOfDays(y,m);

        Console.WriteLine(n);

    }
}

```

=====

45. Find Lowest

45.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

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 { }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Workout44
{
    class UserProgramCode
    {
        public static int getNumberOfDays(int n, int[] a)
        {
            Array.Sort(a);
            return a[0];
        }
    }
}

```

```

    }
}

class Program
{
    static void Main(string[] args)
    {

        int n = int.Parse(Console.ReadLine());
        int[] a = new int[n];
        for (int i = 0; i < n; i++)
        {
            a[i] = int.Parse(Console.ReadLine());
        }

        int op= UserProgramCode.getNumberOfDays(n,a);
        Console.WriteLine(op);

    }
}

```

=====

46. find average:

46. Find Average

Write a program to read an Integer (the size of the List) and the List of Integers and find the average of the numbers as a float value. Print the average.

Print Error Code “Negative numbers present” when inputs other than positive numbers is given.

Include a class UserProgramCode with a static method findAverage which accepts an Integer list. The return type (Float) should return the average value. 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 list, 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 a Float, the average value, 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

2

3

Sample Output 1:

2.5

Sample Input 2:

2

1

-2

Sample Output 2:

Negative numbers present

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace ConsoleApplication18
```

```
{
```



```

class UserProgramCode
{
    public static float compute(int[] array, int size)
    {
        float avg,sum = 0;
        int i;
        foreach (int a in array)
        {
            if (a < 0)
                return -1;
        }
        for (i = 0; i < size; i++)
        {
            sum = sum + array[i];
        }
        avg = sum / size;

        return avg;
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication18
{
    class Program

```

```

{
    static void Main(string[] args)
    {
        UserProgramCode u = new UserProgramCode();
        int n;
        float avg;
        n = int.Parse(Console.ReadLine());
        int[] a = new int[n];
        for (int i = 0; i < n; i++)
        {
            a[i] = int.Parse(Console.ReadLine());
        }
        avg = UserProgramCode.compute(a, n);
        if (avg == -1)
        {
            Console.WriteLine("Negative numbers present");
        }
        else
            Console.WriteLine(String.Format("{0:0.0}", avg));
        }
    }
}

```

=====

47. Validate phone number

47. Validate Phone Number

Write a program to read a phone number as a string input and to verify the phone number using following business rules:

-it should contain only numbers or dashes (-)

-dashes may appear at any position

-should have exactly 10 digits

If the Phone number is valid print “Valid” otherwise print “Invalid”.

Example:

input = 265-265-7777

output = Valid

Include a class UserProgramCode with a static method validatePhoneNumber which accepts a String. The return type (Integer) should return 1 if valid, else return 2.

Create a Class Program which would be used to accept a String, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a String, which corresponds to the phone number.

Output consists of a String, “Valid” if the phone number is valid, else “Invalid”.

Refer sample output for formatting specifications.

Sample Input 1:

265-265-7777

Sample Output 1:

Valid

Sample Input 2:

1111-111-1111

Sample Output 2:

Invalid

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

namespace ConsoleApplication18
{
    class UserProgramCode
    {
        public static int validatephonenummer(string s)
        {
            int len = s.Length;
            int digit = 0, flag = 0;
            char[] a = s.ToCharArray();
            for (int i = 0; i < len; i++)
            {
                if ((a[i] == '-') || char.IsDigit(a[i]))
                {
                    flag++;
                }
            }
            if (flag == len)
            {
                for (int i = 0; i < len; i++)
                {
                    if (char.IsNumber(a[i]))
                    {
                        digit++;
                    }
                }
            }
        }
    }
}
```

```

        }

    }
    if (digit == 10)
        return 1;
    else

        return 2;

    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication18
{

    class Program
    {
        static void Main(string[] args)
        {
            UserProgramCode u = new UserProgramCode();
            string s;
            int result;

```

```

s = Console.ReadLine();
result = UserProgramCode.validatephonenumbers(s);
if(result==1)
    Console.WriteLine("valid");
else if(result==2)
    Console.WriteLine("Invalid");
}

}
}

```

another method:

phone number validation

eg:123-123-1111

```

string s = Console.ReadLine();
Regex re = new Regex(@"([0-9]{3}-[0-9]{3}-[0-9]{4})$");
if (re.IsMatch(s))
{
    Console.WriteLine("vaild");
}
else
{
    Console.WriteLine("invalid");
}
Console.ReadKey();

```

=====

48. check supply

#### 48. Check Supply

The Policy followed by a Company to process Customer Orders is given by the following rules:

Rules:

- (a) If a Customer Order is less than or equal to that in Stock and if the Credit is OK, Supply the required quantity.
- (b) If the Credit is Not OK, then do not Supply. Send him intimation saying "Cannot Supply".
- (c) If the Credit is OK, but the item in Stock is less than the order, Supply what is in Stock. Intimate to him that the balance will be shipped.
- (c) If the Credit is OK and the item in Stock is 0, Output should be "Out Of Stock".

Input1- Stock in hand

Input2- Customer Order Quantity

Input3- Credit (true -OK, false -Not OK)

Output1- Message("Supply", "Cannot Supply", "Out Of Stock", "Balance Will Be Shipped Later")

Example:

Input1: 50

Input2: 5

Input3: true

Output1: Supply

Include a class UserProgramCode with a static method checkSupply which accepts two Integers and a Boolean value. The return type (String) should return a String according to the business rules.

Create a Class Program which would be used to accept two Integers and a Boolean value, and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a 2 integers and a boolean value.

Output consists of a String.

Refer sample output for formatting specifications.

Sample Input 1:

50

5

true

Sample Output 1:

Supply

Sample Input 2:

50

5

false

Sample Output 2:

Cannot Supply

Sample Input 3:

50

55

true

Sample Output 3:

Balance Will Be Shipped Later

Sample Input 4:

0



5

true

Sample Output 4:

Out Of Stock

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

namespace ConsoleApplication18
{
    class UserProgramCode
    {
        public static string checksupply(int n1, int n2,bool value)
        {
            if (value == true && n1 == 0)
            {
                return ("OutOfStock");
            }
            else

                if (value == true && n2 < n1 | n1 == n2)
                {
                    return ("Supply");
                }
            else
```

```
        if (value == true && n1 < n2)
        {
            return ("Balance Will Be Shipped Later");
        }

        else
        {
            if (value == false)
            {
                return "Cannot Supply";
            }
            else
            {
                return "";
            }
        }
    }
}
```

```
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication18
{
    class Program
    {
        static void Main(string[] args)
```

```

{
    UserProgramCode u = new UserProgramCode();
    int n1, n2;
    bool value;
    string output;
    n1 = int.Parse(Console.ReadLine());
    n2=int.Parse(Console.ReadLine());
    value=bool.Parse(Console.ReadLine());
    output = UserProgramCode.checksupply(n1, n2, value);
    Console.WriteLine(output);

}

}
}

```

=====

49. count characters

49.Count Characters

Write a program to count the number of characters present in the given input String array.

Include a class UserProgramCode with static method countCharacters which accepts string array. The return type is a integer value which is the count of characters in the string array.

Create a class Program which would get the input and call the static method countCharacters present in the UserProgramCode.

Input string must contains only the alphabets then return count of characters else return the -1.

If count value is -1 then print "Invalid Input".

Input and Output Format :

Input consists of a integer and String array. Integer represents a size of the array following by the string elements.

Output consists of a integer which is the count of the character from string array.

Sample Input 1:

3

cherry

apple

blueberry

Sample Output 1:

20

Sample Input 2:

2

@aaa

bbb

Sample Output 2:

Invalid Input

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace ConsoleApplication18
```

```

{
class UserProgramCode
{
public static int countcharacters(string[] s)
{
int sum = 0,flag = 0 ;

foreach (string s1 in s)
{
char[] ch = s1.ToCharArray();
foreach (char c in ch)
{
if (char.IsLetter(c))
{
flag++;
}

}

}

foreach (string s1 in s)
{
sum = sum + s1.Length;
}

if(flag==sum)
return sum;

else
return -1;
}
}

```

```
    }  
    }  
}
```

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

```
namespace ConsoleApplication18
```

```
{
```

```
    class Program
```

```
    {
```

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

```
        {
```

```
            UserProgramCode u=new UserProgramCode();
```

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

```
            string[] s=new string[n];
```

```
            int result;
```

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

```
                s[i] = Console.ReadLine();
```

```
            result=UserProgramCode.countcharachters(s);
```

```
            if(result==-1)
```

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

```
            else
```

```
                Console.WriteLine(result);
```

```
        }
```

```
    }
```

```
}  
}
```

another method:

```
static int CountCharacters(List<string> list)
```

```
{
```

```
int count = 0;
```

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

```
{
```

```
char[] c1 = list[i].ToCharArray();
```

```
foreach (char c2 in c1)
```

```
{
```

```
    if (!char.IsLetter(c2))
```

```
    {
```

```
return -1;
```

```
    }
```

```
else
```

```
{
```

```
count++;
```

```
}
```

```
}
```

```
}
```

```
return count;
```

```
}
```

```
}
```

```
}
```

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```

namespace ConsoleApplication18
{
    class UserProgramCode
    {
        public static int countcharacters(string[] s)
        {
            foreach (var item in s)
            {
                char[] ch = item.ToCharArray();
                foreach (var item1 in ch)
                {
                    if (!char.IsLetter(item1))
                    {
                        return -1;
                    }
                }
            }
            int sum = 0, flag = 0;
            int len = 0;
            foreach (var item in s)
            {
                len = item.Length;
                sum = sum + len;
            }
            return sum;
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            string[] s = new string[n];
            int result;
            for (int i = 0; i < n; i++)
                s[i] = Console.ReadLine();
            result = UserProgramCode.countcharacters(s);

            Console.WriteLine(result);
        }
    }
}

```

=====

50. Get Big Diff

50. Get Big Diff



Write a program that reads an integer array and finds the difference between the largest element and smallest element in the array.

Include a class `UserProgramCode` with a static method `getBigDiff` that accepts an integer array and returns an integer.

Create a Class Program which would be used to read the integer array and call the static method present in `UserProgramCode`.

Input and Output Format:

Input consists of  $n+1$  integers. The first integer corresponds to  $n$ , the number of elements in the array.

The next ' $n$ ' integers correspond to the elements in the array.

Output consists of an integer.

Sample Input 1:

4  
10  
3  
5  
6

Sample Output 1:

7

Sample Input 2:

4  
2  
-10  
7  
-2

Sample Output 2:

17

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

namespace ConsoleApplication18
{
    class Program
    {
        static void Main(string[] args)
        {
            UserProgramCode u=new UserProgramCode();
            int n = int.Parse(Console.ReadLine());
            int[] a = new int[n];
            int result;
            for (int i = 0; i < n; i++)
                a[i] = int.Parse(Console.ReadLine());
            result = UserProgramCode.getBigDiff(a);
            Console.WriteLine(result);
        }
    }
}
```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication18
{
    class UserProgramCode
    {
        public static int getBigDiff(int[] a)
        {
            int min = a[0], max = a[0], diff = 0;
            for (int i = 0; i < a.Length; i++)
            {
                if (a[i] >= max)
                {
                    max = a[i];
                }
                if (a[i] <= min)
                {
                    min = a[i];
                }
            }
            diff = max - min;
            return diff;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication18
{
    class Program
    {
        static void Main(string[] args)
    }
}

```

```

    {
        UserProgramCode u = new UserProgramCode();
        int n = int.Parse(Console.ReadLine());
        int[] a = new int[n];
        int result;
        for (int i = 0; i < n; i++)
            a[i] = int.Parse(Console.ReadLine());
        result = UserProgramCode.getBigDiff(a);
        Console.WriteLine(result);
    }
}
class UserProgramCode
{
    public static int getBigDiff(int[] a)
    {
        Array.Sort(a);
        int diff = a[a.Length - 1] - a[0];
        return diff;
    }
}
}

```

=====

Program 51:

#### 51. Calculate the Efficiency

In a company, worker efficiency is determined on the basis of the time required for a worker to complete a particular job. If the time taken by the worker is input, then display the efficiency of the worker.

If time taken is 1 to 3 hours then the worker is said to be "Highly efficient".

If time taken is more than 3 hrs and less than or equal to 4 hours then efficiency level is "Improve speed"

If time taken is more than 4 hours and less than or equal to 5 hours then efficiency level is "Need Training" to improve speed.

If the time taken is more than 5 hours, then the worker has to "Leave the company".

otherwise it should return Invalid Input

Include a class UserProgramCode with static method EfficiencyChecker which accepts float. The return type is String.

Create a class Program which would get the input and call the static method EfficiencyChecker present in the UserProgramCode.

Input output format

The input consists a float.

The output consists the String.

Sample Input 1:

5.0

Sample Output 1:

Need Training

Sample Input 2:

10.5

Sample Output 2:

Leave the company

Sample Input 2:

-2

Sample Output 2:

Invalid Input

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace progm51
```

```

{
    class Program
    {
        static void Main(string[] args)
        {
            decimal inp=Convert.ToDecimal(Console.ReadLine());
            string outp=UserProgramCode.EfficiencyChecker(inp);
            Console.WriteLine(outp);
        }
    }
}

```

```

class UserProgramCode
{
    public static string EfficiencyChecker(decimal a)
    {
        if ((a >= 1) && (a <= 3))
        {
            return "Highly efficient";
        }
        else if ((a > 3) && (a <= 4))
        {

            return "Improve speed";
        }
        else if ((a > 4) && (a <= 5))
        {
            return "Need Training";
        }
        else if (a > 5)

```

```

        {
            return "Leave the company";

        }
        else
        {
            return "Invalid Input";
        }
    }

}
}

```

---

Program 52:

#### 52. Remove Vowels from Even Position

Write code to remove vowels from even position in the string. Return final string as output. Assume the first character is at position 1 in the given string.

Include a class `UserProgramCode` with static method `removeEvenVowels` that accepts the `String`. The return type is a string.

Create a class `Program` which would get the input and call the static method `removeEvenVowels` present in the `UserProgramCode`.

Input output format

Input consists of a string.

Output consists of a string.

Sample Input 1:

commitment

Sample Output 1:

cmmitmnt

Sample Input 2:

rythm

Sample Output 2:

rythm

=====

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace progm52

{

class Program

{

static void Main(string[] args)

{

string inpt;

inpt = Convert.ToString(Console.ReadLine());

string outp=UserProgramCode.removeEvenVowels(inpt);

Console.WriteLine(outp);

}

}

class UserProgramCode

{



```

public static string removeEvenVowels(string a)
{
    int maxi=a.Length;

    for (int i = 1; i < maxi; i+=2)
    {
        if ((a[i] == 'a') || (a[i] == 'e') || (a[i] == 'i') || (a[i] == 'o') || (a[i] == 'u'))
        {
            a=a.Remove(i,1);
            maxi--;
            i++;
        }
    }
    return a;
}
}

```

---

Program 53:

### 53.Count Between Numbers

Write a program to find the count of elements in the range [l, h] (both inclusive) in an integer array. l corresponds to the lower value and h corresponds to the higher value.

Include a class UserProgramCode with a static method countBetweenNumbers which accepts an integer array and two other integers ( l and h). The return type is int. If there are negative numbers in the array or if the value of l or h is negative, return -1.

Create a Class Program which would be used to read the inputs and call the static method present in UserProgramCode.

If the method returns -1, then print "Negative value Present"

Input and Output Format:

Input consists of n+3 integers.

The first integer corresponds to n, the number of elements in the array.

The next 'n' integers correspond to the elements in the array.

The last integers correspond to the lower value and higher value.

Output consists of an integer or a string.

Refer sample output for formatting specifications.

Sample Input 1:

9  
2  
13  
6  
19  
25  
65  
34  
1  
20  
5  
20

Sample Output 1:

4

Sample Input 2:

2

3

-5

2

3

Sample Output 2:

Negative value Present

=====--

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace progm53
```

```
{
```

```
    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 l = Convert.ToInt32(Console.ReadLine());
int h = Convert.ToInt32(Console.ReadLine());
int output=UserProgramCode.countBetweenNumbers(a, l, h);
if (output != -1)
{
    Console.WriteLine(output);
}
else
{
    Console.WriteLine("Negative value Present");
}
}
}

```

class UserProgramCode

```

{
    public static int countBetweenNumbers(int[] inp,int x,int y)
    {
        int max1=inp.Length;
        int count = 0;
        for (int i = 0; i < max1; i++)
        {
            if ((inp[i] < 0) || (x < 0) || (y < 0))
            {
                count= -1;
            }
            else
            {
                if ((inp[i] >= x) && (inp[i] <= y))
                {

```

```

        count++;
    }
}
}
return count;

}
}
}

```

Program 54:

#### 54. 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 = inputstr1.Remove(0, z);

        ans = inputstr1 + inputstring2;
    }

    return ans;

}

}

}

```

---

Program 55:

55.SortList

Write a code to sort the given array of integers in ascending order.

Business Rules:

Only positive numbers should be given as inputs to the integer array.

Include a class UserProgramCode with static method sortList which accepts interger array

The return type is a interger array. If the input consists of negative numbers,return -1. If the input array size is 0, return -1.

Create a class Program which would get the input and call the static method sortList present in the UserProgramCode.

If the sortList method returns -1 print "Invalid Input".

If the sortList method returns -2 print "Input is Empty".

Input Output Format:

Input consists of a n+1. n represents the size of the array followed by the elements in the array.



Output consists of a array which is sorted in ascending order.

Sample Input 1:

3

45

12

36

Sample Output 1 :

Sorted Array :

12

36

45

Sample Input 2:

4

-1

56

89

45

Sample Output 2 :

Invalid Input

=====

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

```
namespace prog55
```

$$\{$$

```
class UserProgramCode
```

 $\{$ 

```
public static int[] sortList(int[] a)
```

 $\{$ 

```
int temp;
```

```
int maxi = a.Length;
```

```
if (maxi == 0)
```

 $\{$ 

```
a[0] = -1;
```

}

else

$$\{$$

```
for (int i = 0; i < maxi; i++)
```

 $\{$ 

```
for (int j = 0; j < maxi; j++)
```

 $\{$ 

```
if (a[i] < 0)
```

$$\{$$

```
a[0] = -1;
```

$$\}$$

else

```
        {  
            if (a[i] < a[j])  
            {  
                temp = a[i];  
                a[i] = a[j];  
                a[j] = temp;  
            }  
        }
```

```
    }  
}  
}
```

```
}
```

```
    return a;
```

```
}
```

```
}
```

```
class Program
```

```
{
```

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

```
{
```

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

```
    int[] a=new int[n];
```

```
    int[] output=new int[n];
```

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

```

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

}

output=UserProgramCode.sortList(a);


if (output[0] != -1)
{
    for (int i = 0; i < n; i++)
    {
        Console.WriteLine(output[i]);
    }
}
else
{
    Console.WriteLine("Invalid Input");
}

}

}

```

---

PROGRAM 56.

56. 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 && c)  
            {  
                if ((str.Length >= 6) && (str.Length <= 20))  
                {  
                    return 1;  
                }  
            }  
            return -1;  
        }  
    }  
}
```

```
-----  
class UserProgramCode
```

```
{  
    public static string su(string s)  
    {  
        if (Regex.IsMatch(s, @"^(?=[\d])(?=[a-zA-Z])(?=[@#$])([a-zA-z0-9$#@]{6,20}))")  
        {
```

```
        return "valid";
    }
    else
        return "invalid";

}
}
```

=====

## PROGRAM 57.

### 57. Longest Word

Write a Program which finds the longest word from a sentence. Your program should read a sentence as input from user and return the longest word. In case there are two words of maximum length return the word which comes first in the sentence.

Include a class UserProgramCode with a static method getLargestWord which accepts a string. The return type is a string that corresponds to the largest word in the sentence.

Create a Class Program which would be used to accept a input string and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a string with a maximum size of 100 characters.

Output consists of a single string.



Refer sample output for formatting specifications.

Sample Input 1:

Welcome to the world of Programming

Sample Output 1:

Programming

Sample Input 2:

ABC DEF

Sample Output 2:

ABC

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace level1_57
```

```
{
```

```
    class Program
```

```
    {
```

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

```

        {
            string s = Console.ReadLine();

            string c;

            c=UserProgramCode.getLargestWord(s);

            Console.WriteLine(c);
        }
    }

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace level1_57
{
    class UserProgramCode
    {
        public static string getLargestWord(string str)
        {
            int l=0,max=0,ind=-1;

            string[] s=new string[100];

            s = str.Split(' ');

            for (int i = 0; i < s.Length; i++)
            {
                l = s[i].Length;

                if (max < l)
                {
                    max = l;

                    ind = i;
                }
            }
        }
    }
}

```

```

        }
        return s[ind];

    }
}
}

=====

```

## PROGRAM 58.

58. 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;

```

```

        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;
        return d;

    }
}
}

```

=====

## PROGRAM 59.

### 59. 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;
}
}

```

=====

## PROGRAM 60.

### 60. Three Digits

Write a program to read a string and check if the given string is in the format "CTS-XXX" where XXX is a three digit number.

Include a class UserProgramCode with a static method validatestrings which accepts a string and returns an integer.. The function returns 1 if the string format is correct, else returns -1.

Create a Class Program which would be used to accept a String and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of a string.

Output consists of a string (Valid or Invalid).

Refer sample output for formatting specifications.

Sample Input 1:

CTS-215

Sample Output 1:

Valid

Sample Input 2:

CTS-2L5

Sample Output 2:

Invalid

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

namespace level1_60
{
    class Program
    {
        static void Main(string[] args)
        {
            int x;
            string s = Console.ReadLine();
            x = UserProgramCode.validatestrings(s);
            if (x == 1)
            {
                Console.WriteLine("valid");
            }
            else
            {
                Console.WriteLine("Invalid");
            }
        }
    }
}

using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;

namespace level1_60
{
    class UserProgramCode
    {
        public static int validatestrings(string str)
        {
            int output1 = 0;
            Regex reg = new Regex(@"^([C]+[T]+[S]+[-]+([0-9]{3}))$");
            if (reg.IsMatch(str))
            {
                output1 = 1;
            }

            else
            {
                output1 = -1;
            }
            return output1;
        }
    }
}
```

=====

### 61.61.File Extension

Write a program to read a file name as a string and find out the file extension and return it as output. For example, the file sun.gif has the extension gif.

Include a class UserProgramCode with a static method fileIdentifier which accepts a string. The return type (string) should return the extension of the input string (filename).

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 file name.

Output consists of a string(extension of the input string (filename)).

Refer sample output for formatting specifications.

Sample Input 1:

sun.gif

Sample Output 1:

gif

```
class Program
{
    static void Main(string[] args)
    {
        string s;
```

```

        s = Console.ReadLine();
        userprogramcode obj = new userprogramcode();
        Console.WriteLine(obj.fileIdentifier(s));

    }
}
public class userprogramcode
{
    public string fileIdentifier(string s)
    {
        string[] str;
        str=s.Split('.');
        return str[1];
    }
}

```

=====

## 6262.Number Validation

Write a program to read a string of 10 digit number and to check whether the string contains a 10 digit number in the format XXX-XXX-XXXX where 'X' is a digit.

Include a class UserProgramCode with a static method validateNumber which accepts a string as input and returns an Integer .

The method returns 1 if the string meets the above specified format . If the string input does not meet the specified format the method returns -1.

Create a class Program which would get the input as a String and call the static method validateNumber present in the UserProgramCode.

Input and Output Format:

Input consists of a string.

Output is a string specifying the given string is valid ("Valid number format") or not ("Invalid number format") .

Refer sample output for formatting specifications.

Sample Input 1:

123-456-7895

Sample Output 1:

Valid number format

Sample Input 2:

-123-12344322

Sample Output 2:

Invalid number format

```
class Program
{
    static void Main(string[] args)
    {
        int f = 0;
```

```

string s;
s = Console.ReadLine();
userprogramcode obj = new userprogramcode();
f=obj.validatenumbers(s);
if(f==1)
    Console.WriteLine("Valid number format");
if(f==-1)
    Console.WriteLine("Invalid number format");

}
}
public class userprogramcode
{
    public int validatenumbers(string s)
    {
        if (Regex.IsMatch(s, @"^\d{3}[-]\d{3}[-]\d{4}$"))
        {
            return 1;
        }
        else
            return -1;
    }
}

```

=====

### 6363.Sum of cubes and squares of elements in an 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

Sample Output 1:

208

```
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;
    }
}

```

```

=====
=====

```

#### 6464.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;
    }
}

```

=====

6565.Difference between two dates in days

Get two date strings as input and write code to find difference between two dates in days.

Include a class UserProgramCode with a static method getDateDifference which accepts two date strings as input.

The return type of the output is an integer which returns the difference between two dates in days.

Create a class Program which would get the input and call the static method getDateDifference 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:

2012-03-12

2012-03-14

Sample Output 1:

2

Sample Input 2:

2012-04-25

2012-04-28

Sample Output 2:

3

```
class Program
{
    static void Main(string[] args)
    {
        string s,s1;
```

```

        s = Console.ReadLine();
        s1 = Console.ReadLine();
        userprogramcode obj = new userprogramcode();
        Console.WriteLine(obj.getDateDifference(s,s1));

    }
}
public class userprogramcode
{
    public string getDateDifference(string s,string s1)
    {
        string fm="yyyy-MM-dd";
        DateTime dt1,dt;
        DateTime.TryParseExact(s,fm,null,System.Globalization.DateTimeStyles.None,out dt);
        DateTime.TryParseExact(s1, fm, null, System.Globalization.DateTimeStyles.None, out dt1);

        return Convert.ToString((dt1-dt).Days);
    }
}

```

=====

## 66) 66.Count Sequential Characters

Get a string as input and write code to count the number of characters which gets repeated 3 times consecutively and return that count (ignore case).

Include a class UserProgramCode with a static method countSequentialChars which accepts a string as input and return type is an integer.

The method returns the repeat count. If no character gets repeated 3 times consecutively the method returns -1.

Create a class Program which would get the input and call the static method countSequentialChars present in the UserProgramCode.

If the method returns -1, print 'No Repeated Words Found'.

Input and Output Format:

Input consists a string.

Refer sample output for formatting specifications.

Sample Input 1:

abcXXXabc

Sample Output 1:

1

Sample Input 2:

aaxxyzAAx

Sample Output 2:

No Repeated Words Found

```
using System;
```

```
using System.Collections.Generic;si
```



```
using System.Linq;
using System.Text;

namespace ConsoleApplication24
{
    class Program
    {
        static void Main(string[] args)
        {
            string s = Console.ReadLine();
            int a = UserProgramCode.countSequentialChars(s);
            if(a== -1)
                Console.WriteLine("No Repeated Words Found");
            else
                Console.WriteLine(a);
            Console.ReadLine();

        }
    }
    class UserProgramCode
    {
        public static int countSequentialChars(string s)
        {

            int l = s.Length;
            string[] st = new string[50];

            for (int k = 0; k < l; k++)
            {
                st[k] = s.Substring(k, 1);
```

```
}
```

```
int count = 0;
```

```
int c=0;
```

```
for (int k = 0; k < l - 1; k++)
```

```
{
```

```
    if (st[k] == st[k+1])
```

```
        count++;
```

```
    else
```

```
        count = 0;
```

```
    if (count == 2)
```

```
        c++;
```

```
}
```

```
if(c==0)
```

```
return -1;
```

```
else
```

```
    return c;
```

```
    }  
    }  
}
```

another method:

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

```
namespace count_sequential  
{  
    class Userprogramcode  
    {  
        public static int countSequentialChars(string s)  
        {  
            int count=0;  
            char[] c = s.ToCharArray();  
            for (int i = 0; i < c.Length-2; i++)  
            {  
                if(c[i]==c[i+1])  
                {  
                    if(c[i+1]==c[i+2])  
                    {  
                        count++;  
                    }  
                }  
            }  
        }  
    }  
}
```

```

    }
    if (count == 0)
    {
        return -1;
    }
    else
    return count;

}
}
}

```

=====

#### 67) 67. Boundary Average

Given an int array as input, write a program to compute the average of the maximum and minimum element in the array.

Include a class UserProgramCode with a static method “getBoundaryAverage” that accepts an integer array as argument and return a average of max and min value.

Create a class Program which would get the input array and call the static method getBoundaryAverage present in the UserProgramCode.

Input and Output Format:

The first line of the input consists of an integer n, that corresponds to the size of the array.

The next n lines consist of integers that correspond to the elements in the array.

Assume that the maximum number of elements in the array is 10.

Output consists of a single float value that corresponds to the average of the max and min element in the array.

Output is displayed correct to 1 decimal place.

Sample Input :

6

3

6

9

4

2

5

Sample Output:

5.5

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace ConsoleApplication24
```

```
{
```

```
    class Program
```

```

{
    static void Main(string[] args)
    {

        int n=Convert.ToInt32(Console.ReadLine());
        int[] s =new int[n];
        for(int i=0;i<n;i++)
            s[i] =Convert.ToInt32(Console.ReadLine()) ;
        double a = UserProgramCode.getBoundaryAverage(s);
        Console.WriteLine(a);
        Console.ReadLine();

    }
}

class UserProgramCode
{
    public static double getBoundaryAverage(int[] a)
    {

        double d,e;
        d=((a.Max()+a.Min())/2.0);
        e = Math.Round(d, 1);

        return e;
    }
}

```

```
    }  
  }  
}
```

=====

#### 68) 68.String Reversal

Write a program to reverse each word in the given string.

Include a class `UserProgramCode` with a static method “reverseString” that accepts a string argument and returns a string.

If string contains any special characters then return "-1".

Create a class `Program` which would get a string as input and call the static method `reverseString` present in the `UserProgramCode`.

If the method returns -1, then print 'Invalid Input'.

Input and Output Format:

Input consists of a string.

Output consists of a string.

Sample Input 1:

hai hello

Sample Output 1:

iah olleh

Sample Input 2:

how !#\$

Sample Output 2:

Invalid Input

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {

            string s = Console.ReadLine();

            string a = UserProgramCode.reverseString(s);
            if (a == "-1")
                Console.WriteLine("Invalid Input");
            else
                Console.WriteLine(a);
            Console.ReadLine();

        }
    }
}
```



```

class UserProgramCode
{
    public static string reverseString(string a)
    {
        int l=a.Length;

        if (a.Any(ch => !(Char.IsLetterOrDigit(ch) || char.IsWhiteSpace(ch))))
            return "-1";

        StringBuilder sb = new StringBuilder();
        char[] c;
        string[] s;
        s=a.Split(' ');
        for(int i=0;i<s.Length;i++)
        {
            c= s[i].ToCharArray();

            Array.Reverse(c);
            sb.Append(c);
            sb.Append(" ");
        }

        return sb.ToString();
    }
}

```

```
}  
}
```

=====

#### 69) 69. Find Occurrence

Write a method to find the occurrence (number of times) of a given character in a given input string.

Include a class `UserProgramCode` with a static method `findOccurrence` which accepts a string and character as input and returns an integer.

Business Rules:

1. Search criteria is irrespective of cases.
2. The input string should consist of only alphabets, no special characters or numbers should be there in the string. If present, the method returns -1.

Create a class `Program` which would get the input and call the static method `findOccurrence` present in the `UserProgramCode`.

If the method returns -1, print 'Invalid Input'.

Input and Output format :

Input consists of string and character.

Refer sample output for formatting specifications.

Sample Input 1 :

HELLO friends Welcome to CSharp wonderful world

L

Sample Output 1 :

5

Sample Input 2 :

Gr8...I am fine.

8

Sample Output 2 :

Invalid Input

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {

            string s = Console.ReadLine();
            char c = Convert.ToChar(Console.ReadLine());

            int a = UserProgramCode.findOccurence(s,c);
            if (a == -1)
                Console.WriteLine("Invalid Input");
            else
                Console.WriteLine(a);
        }
    }
}
```

```
Console.ReadLine();
```

```
}
```

```
}
```

```
class UserProgramCode
```

```
{
```

```
    public static int findOccurence(string a,char b)
```

```
    {
```

```
        int count = 0;
```

```
        if (a.Any(ch => !(Char.IsLetter(ch)||Char.IsWhiteSpace(ch))))
```

```
            return -1;
```

```
        foreach (char e in a)
```

```
        {
```

```
            if (e==char.ToUpper(b)||e==char.ToLower(b))
```

```
                count++;
```

```
        }
```

```
        return count;
```

```
}
```

```
}
```

```
}
```

another method:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Text.RegularExpressions;
```

```
namespace find_occurences
```

```
{
```

```
    class Userprogramcode
```

```
    {
```

```
        public static int findOccurence(string a, char b)
```

```
        {
```

```
            char[] ch = a.ToCharArray();
```

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

```
            {
```

```
                if (!char.IsLetter(ch[i]))
```

```
                {
```

```
                    return -1;
```

```
                }
```

```
            }
```

```
            int count = 0;
```

```
            a.ToLower();
```

```

char[] c = a.ToCharArray();
for (int i = 0; i < c.Length; i++)
{
    if (c[i] == b)
    {
        count++;
    }
}

return count;
}
}
}

```

=====

70) 70.Calculate Commission

Write a program to calculate the commission on given sales as per the following policy.

Include a class UserProgramCode with a static method calculateCommission which accepts a float as input.

Create a class Program which would get the input and call the static method calculateCommission present in the UserProgramCode.

If the method returns -1, then print 'Invalid Input'.

If sales is less than Rs. 10000/- no commission.

If sales is between Rs. 10000/- to Rs. 25000/- commission is 10% of sales.

If sales is more than Rs. 25000/- then commission is Rs. 500/- plus 8% of sales amount.

Business Rule :

1. If input is negative number then the method calculateCommission returns -1.
2. Otherwise return a calculated commission.

Input and Output format :

Input consists of float.

Refer sample output for formatting specifications.

Sample Input 1 :

11000

Sample Output 1:

1100

Sample Input 2 :

-1000

Sample Output 2 :

Invalid Input

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```

namespace ConsoleApplication26
{
    class Program
    {
        static void Main(string[] args)
        {

            float s = float.Parse((Console.ReadLine()));

            int a = UserProgramCode.calculateCommission(s);
            if (a == -1)
                Console.WriteLine("Invalid Input");
            else
                Console.WriteLine(a);
            Console.ReadLine();

        }
    }
    class UserProgramCode
    {
        public static int calculateCommission(float a)
        {
            double com;
            double r = Convert.ToDouble(a);

            if (r < 0)
            {
                return -1;
            }
        }
    }
}

```



```

else if (r < 10000)
{
    return 0;
}
else if (r >= 10000 && r <= 25000)
{
    com = r * 1.0 / 10.0;
    return Convert.ToInt32(com);

}

else if (r > 25000)
{
    com = 500 + (r * 8.0/ 100.0);
    return Convert.ToInt32(com);

}
return -1;

}
}
}

```

=====

Program 71:

#### 71.Sort Array Element

Write a method to sort input string array by the length of each element. If word length is same then sort these 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 next '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);
    return a;
}
}

```

another method:

```

class userProgramCode
{
    public static string[] sortArrayElement(string[] a)
    {
        string[] c=new string[1];

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

```

```

        if(!Regex.IsMatch(a[i],@"^([A-Za-z0-9])$"))
        {

            c[0] = "-1";

            return c;

            System.Environment.Exit(0);

        }
    }

    var q = from z in a
            orderby z.Length
            select z;

    string[] s = q.ToArray();

    return s;

}

}

}

```

---

Program 72:

## 72. Shortest Word Length

Given a string array as input, write a program to find the length of the shortest word in the array..

Create a class named UserProgramCode that has the following static method

```
public static int shortestWordLength(string[] input1)
```

Create a class named Program that accepts a string array as input 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 string array.

The next 'n' lines of the input consists of strings that correspond to the elements in the string array.

Output consists of a single integer that corresponds to the length of the shortest word in the array.

Sample Input :

```
3
cherry
apple
blueberry
```

Sample Output :

```
5
```

```
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];
            for (int i = 0; i < a.Length; i++)
                a[i] = Console.ReadLine();
            int r = userProgramCode.shortestWordLength(a);
            Console.WriteLine(r);
            Console.ReadLine();
        }
    }
}

class userProgramCode
{
    public static int shortestWordLength(string[] a)
    {
        int min=1000;
        for (int i = 0; i < a.Length; i++)
        {
            if (i == 0)
                min = a[i].Length;
            else
            {
                if (min > a[i].Length)
                    min = a[i].Length;
            }
        }
    }
}

```

```

    }
    return min;
}
}
}

```

another method:

```

public static int shortestWordLength(string[] a)
{
    int min =a[0].Length;
    for (int i = 0; i < a.Length; i++)
    {
        if (min>a[i].Length)
        {
            min=a[i].Length;
        }

    }
    return min;
}
}
}

```

=====

Program 73:

73.Shipping Cost



Write a program to compute the Cost of Book Shipping. The Shipping Cost is computed according to the shipping type and the package weight. The shipping rate is given below.

Shipping types - Weight Rate (bahts/gram)

Regular for first 2000 - 0.25 (basic charge)

Regular exceeding 2000 - 0.35 for each

Express uses the same rate as Regular + 50 bahts fee

Note that the Shipping cost is computed from the possible valid minimum rate.

Input1- Weight in grams

Input2- Type of delivery ('R' Regular and 'X' Express)

Create a class named UserProgramCode that has the following static method

```
public static float CalcShippingCost(float 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:

The first line of the input consists of a float that corresponds to the weight in grams.

The second line of the input consists of a character ('R' or 'X') that corresponds to the type of service.

Output consists of a single float that corresponds to the shipping cost. Output is displayed correct to 2 decimal places.

Sample Input:

4500

R

Sample Output:

1375.00

```
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)
        {
            float f=float.Parse(Console.ReadLine());
```

```

        char c=Convert.ToChar(Console.ReadLine());
        float p=userProgramCode.CalcShippingCost(f,c);
        Console.WriteLine(p.ToString("F"));
        Console.ReadLine();
    }
}
class userProgramCode
{
    public static float CalcShippingCost(float i, char c)
    {
        float p;
        if (c == 'X')
            i += 50;
        if (i > 2000)
        {
            float d = i - 2000;
            p =Convert.ToSingle(2000 * 0.25);
            p +=Convert.ToSingle( (d * 0.35));
        }
        else
            p = Convert.ToSingle(i * 0.25);
        return p;
    }
}

```

---

Program 74:

74.Strong Number

Write a program to find whether the given integer input is a strong number or not. If the sum of each digits factorial is the same as the given input value then it is a strong number.

If the Input1 is strong number then print "Input1 is a Strong Number" where Input1 is the input integer value. (Refer Example)

Business rule:

1) If the Input1 value is not a strong number then print "Sum of all digits factorial is XX" where XX is the total of each digits factorial value.

2) Print "Invalid Input" when given input number is a negative number.

Example:1

Input1: 145

$1!+4!+5! = 1+24+120 = 145$

Output1: 145 is a Strong Number

Example:2

Input1: 25

$2!+5! = 2+120 = 122$

Output1: Sum of all digits factorial is 122

Create a class named UserProgramCode that has the following static method

`public static String checkStrongNumber(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 a single integer.

Output is a string.

Sample Input 1:

145

Sample Output 1:

145 is a Strong Number

Sample Input 2:

25

Sample Output 2:

Sum of all digits factorial is 122

```
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 r = userProgramCode.checkStrongNumber(n);
            Console.WriteLine(r);
            Console.ReadLine();
        }
    }
    class userProgramCode
    {
        public static String checkStrongNumber(int i)
        {
            if (i < 0)
                return "Invalid Input";
```

```

int t, r, f = 0;
t = i;
while (t > 0)
{
    int fact = 1;
    r = t % 10;
    for (int j = 1; j <= r; j++)
    {
        fact *= j;
    }
    f += fact;
    t /= 10;
}
if (f == i)
    return f + " is a Strong Number";
else
    return "Sum of all digits factorial is " + f;

}
}
}

```

Program 75:

75.Print Digit Sum

Write a program that accepts a string input and finds the sum of all numeric digits present in the string.

Example 1:

input : abc12de4

output : 7

Example 2:

input : udjc&23er

output : -1

Business Rules :

1. If the given input string contains any special characters, then print -1.
2. f the given input string contains no numbers,then print -2.

Create a class named UserProgramCode that has the following static method

```
public static int getdigits(string 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 a String

Output is an integer.

Sample Input 1:

abc12de4

Sample Output 1:



7

Sample Input 2:

udjc&23er

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)
        {
            string s=Console.ReadLine();
            int i=userProgramCode.getdigits(s);
            Console.WriteLine(i);
            Console.ReadLine();
        }
    }
}
```

```

class userProgramCode
{
    public static int getdigits(string i)
    {
        int s=0;
        for (int j = 0; j < i.Length; j++)
        {
            if ((char.IsLetterOrDigit(i[j])))
            {
                if (char.IsDigit(i[j]))
                    s += (Convert.ToInt32(i[j])-48);
            }
            else
                return -1;
        }
        if (s == 0)
            return -2;
        else
            return s;
    }
}

```

=====

## 76.76.Special Characters

Write a program that accepts a string input and removes all the alphabetic characters from input and stores only the special characters and digits.

Note: Special characters are #, \$, %, &

Business Rules :

1. if the given input string contains no numbers or special characters, then print -1.

Create a class named UserProgramCode that has the following static method

```
public static string getSpecialChar(string 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 a string.

Output is either '-1' or the processed string.

Sample Input 1:

cogniz\$#45Ant

Sample Output 1:

\$#45

Sample Input 2:

Treasure

Sample Output 2:

-1

```
//program.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
namespace trial
{
    class Program
    {
        static void Main(string[] args)
        {
            String str = Console.ReadLine();
            String res = UserProgramCode.getSpecialChar(str);
            Console.WriteLine(res);
        }
    }
}
```

```
//UserProgramCode.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace trial
{
```

```

class UserProgramCode
{
    public static string getSpecialChar(string input1)
    {
        int num = 0;
        StringBuilder sb=new StringBuilder();
        int sp = 0;
        int len = input1.Length;
        for (int i = 0; i < len; i++)
        {
            char c = input1[i];
            if (char.IsDigit(c))
            {
                num++;
                sb.Append(c);
            }
            if (c == '#' || c == '$' || c == '%' || c == '&')
            {
                sp++;
                sb.Append(c);
            }
            //else if (!char.IsLetter(c))
            //{
            //    sb.Append(c);
            //}
        }
        if (num == 0 || sp == 0)
        {
            return "-1";
        }
    }
}

```

```

        else
            return sb.ToString();

    }
}
}

```

=====

77.

#### 77.Calculate Frequency

Given two string inputs input1 and input2, write a program to find the number of times the complete string in input1 occurs in input2 and print the count. Ignore case sensitiveness in the input strings.

Business Rules:

- 1) If input1 has repeated words, print -1.
- 2) If the count of occurrence is zero then print -2 .

Example1:

input1: A good place

input2: It is a good place to be in and a good place to have fun.

output: 2

Example:

input1 :Does he have to have a car ?

input2: Yes he should.

output: -1

Create a class named UserProgramCode that has the following static method

```
public static int calcFrequency(string input1, string input2)
```

Create a class named Program that accepts the input and calls the static method present in the UserProgramCode.

Input and Output Format:

Input consists of 2 strings.

Output consists of an integer.

Sample Input :

A good place

It is a good place to be in and a good place to have fun.

Sample Output :

2

```
//program.cs
```

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace oops2
```

```
{
```

```
    class Program
```

```
    {
```

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

```

    {
        String input1 = Console.ReadLine();
        String input2 = Console.ReadLine();
        int ret = UserProgramCode.calcFrequency(input1, input2);
        Console.WriteLine(ret);
        //Console.WriteLine(input1.ToLower());
    }
}

}

//UserProgramCode.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace oops2
{
    class UserProgramCode
    {
        public static int calcFrequency(string input1, string input2)
        {
            StringBuilder sb = new StringBuilder();
            int val = 0;
            String[] arr = input1.Split(' ');
            foreach (String a in arr)
            {
                sb.Append(a.ToLower());
            }
            int len = arr.Length;

```



```

for (int i = 0; i < len; i++)
{
    String s = arr[i];
    for (int j = i + 1; j < len; j++)
    {
        if (s.Equals(arr[j]))
        {
            val++;
            return -1;
        }
    }
}
if (val == 0)
{
    int count = 0;

    StringBuilder sb1 = new StringBuilder();
    String linput1 = input1.ToLower();
    String linput2 = input2.ToLower().Replace('.', ' ');
    // Console.WriteLine(linput2);
    String[] l = linput2.Split(' ');
    foreach (String a in l)
    {
        sb1.Append(a);
    }
    len = sb.Length;
    int len2 = sb1.Length;
    // Console.WriteLine(len + "" + len2);
    for (int i = 0; i < len2; i++)
    {

```

```

        if(i<len2-len)
        {
            // Console.WriteLine(i+", "+len);
            // Console.WriteLine(sb);
            // Console.WriteLine(sb1);

            String sub = sb1.ToString().Substring(i, len);
            if (sub.Equals(sb.ToString()))
            {
                count++;
            }
        }

    }

    if (count == 0)
        return -2;
    else
        return count;
}

return 0;
}
}

=====
=====

```

## 78.78.Form String

Given a String array and an int 'n', write a program to perform the following operations:

- 1) Pick nth character from each String element in the String array and form a new String.
- 2) If nth character not available in a particular String in the array consider \$ as the character.

3) Print the new String.

Business Rules :

1. If there are any special characters in the input strings, then print -1.

Create a class named UserProgramCode that has the following static method

```
public static String formString(String[] 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 'k' that corresponds to the number of elements in the string array.

The next 'k' lines of the input consists of strings that correspond to the elements in the string array.

The next line of the input consists of an integer that corresponds to n.

Refer sample output for formatting specifications.

Sample Input :

4

ABC

XYZ

EFG

MN

3

Sample Output :

CZG\$

```
//program.cs
```

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace oops2
```

```
{
```

```
    class Program
```

```
    {
```

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

```
        {
```

```
            int k,n;
```

```
            k = int.Parse(Console.ReadLine());
```

```
            String[] arr = new String[k];
```

```
            for (int i = 0; i < k; i++)
```

```
            {
```

```
                arr[i] = Console.ReadLine();
```

```
            }
```

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

```
            String ret = UserProgramCode.formString(arr, n);
```

```
            Console.WriteLine(ret);
```

```
    }  
}
```

```
}
```

```
//UserProgramCode.cs
```

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace oops2
```

```
{
```

```
    class UserProgramCode
```

```
    {
```

```
        public static string formString(string[] input1, int input2)
```

```
        {
```

```
            int len;
```

```
            StringBuilder sb = new StringBuilder();
```

```
            foreach (String a in input1)
```

```
            {
```

```
                if (a.Contains('#') || a.Contains('&') || a.Contains('%') || a.Contains('$'))
```

```
                    return "-1";
```

```
            else
```

```
            {
```

```
                len = a.Length;
```

```
                // Console.WriteLine(len + "," + input2);
```

```
                if (input2 > len)
```

```
                    sb.Append('$');
```

```

        else
        {
            // Console.WriteLine(a);

            String c = a.Substring(input2-1,1);
            // Console.WriteLine(c);

            sb.Append(c);
        }
    }
}

return sb.ToString();
}
}
}

=====
=====

```

## 79.79.Word Form

Write a program that accepts an integer input and displays the given number in word form. The word form should include only billions, millions, thousands, hundreds wherever applicable. The starting alphabet of each word should be in capital except the word "and".

Business Rules:

1) If the given integer is negative convert that to a positive number and append "Minus" before the word then display the result.

Create a class named UserProgramCode that has the following static method

```
public static string wordForm(int number)
```

Create a class named Program that accepts the input and calls the static method present in the UserProgramCode.

Input and Output Format:

Input consists of an integer.

Output is a string.

Sample Input 1:

364576567

Sample Output1:

Three Hundred and Sixty Four Million Five Hundred and Seventy Six Thousand Five Hundred and Sixty Seven

Sample Input 2:

-1234

Sample Output 2:

Minus One Thousand Two Hundred and Thirty Four

//program.cs

using System;

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

namespace oops2
{

    class Program
    {
        static void Main(string[] args)
        {
            int num;
            num = int.Parse(Console.ReadLine());
            String ret = UserProgramCode.wordForm(num);
            Console.WriteLine(ret);
        }
    }
}
```

```
//UserProgramCode.cs
```

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

namespace oops2
{
    class UserProgramCode
```



```

{
    public static string wordForm(int n)
    {
        return _toText(n, true);
    }
    private static string _toText(long n, bool isFirst = false)
    {
        string result;
        if (isFirst && n == 0)
        {
            result = "Zero";
        }
        else if (n < 0)
        {
            result = "Negative " + _toText(-n);
        }
        else if (n == 0)
        {
            result = "";
        }
        else if (n <= 9)
        {
            result = new[] { "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine" }[n -
1] + " ";
        }
        else if (n <= 19)
        {
            result = new[] { "Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen",
"Seventeen", "Eighteen", "Nineteen" }[n - 10] + (isFirst ? null : " ");
        }
        else if (n <= 99)

```

```

    {
        result = new[] { "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"
    }[n / 10 - 2] + (n % 10 > 0 ? "-" + _toText(n % 10) : null);
    }
    else if (n <= 999)
    {
        result = _toText(n / 100) + "Hundred " + _toText(n % 100);
    }
    else if (n <= 999999)
    {
        result = _toText(n / 1000) + "Thousand " + _toText(n % 1000);
    }
    else if (n <= 999999999)
    {
        result = _toText(n / 1000000) + "Million " + _toText(n % 1000000);
    }
    else
    {
        result = _toText(n / 1000000000) + "Billion " + _toText(n % 1000000000);
    }
    if (isFirst)
    {
        result = result.Trim();
    }
    return result;
}
}
}

```

---

80.80.Repeat Characters

Write a program to repeat the string multiple times provided with the below limitations.

- a. If Input1 string length is five or less than five, then the first three characters should be repeated based on Input2 value.
- b. If the Input1 string length is more than five then the last three characters should be repeated based on Input2 value

Business Rules :

1. If the length of Input1 is less than 3, then print 'Input value is insufficient'
2. If the Input2 value is negative, then print 'Invalid Input'
3. If the Input2 value is greater than 10, then print 'Input value is too long'

Example 1:

Input1: Price

Input2: 3

Output : PriPriPri

Example 2:

Input1: Sunday

Input2: 4

Output: daydaydayday

Example 3:

Input1: So

Input2: 5

Ouput: Input value is insufficient

Create a class named UserProgramCode that has the following static method

```
public static string repeatManipulateString(string 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 a string.

The second line of input consists of an integer.

Output is a string. Refer sample output and business rules for output formatting specifications.

Sample Input:

Price

3

Sample Output:

PriPriPri

```
//program.cs
```

```
using System;
```

```
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;

namespace oops2
{

    class Program
    {
        static void Main(string[] args)
        {
            String val = Console.ReadLine();
            int num = int.Parse(Console.ReadLine());
            String ret = UserProgramCode.repeatManipulateString(val, num);
            Console.WriteLine(ret);
        }
    }
}

```

```
//UserProgramCode.cs
```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace oops2
{
    class UserProgramCode
    {
        public static string repeatManipulateString(string input1, int input2)
        {

```

```
StringBuilder sb = new StringBuilder();
int len = input1.Length;
if (len < 3)
    return "Input value is insufficient";
if (input2 < 0)
    return "Invalid Input";
if (len <= 5)
{
    for (int i = 0; i < input2; i++)
    {
        sb.Append(input1.Substring(0,3));
    }
    return sb.ToString();
}
else if (len > 10)
{
    return "Input value is too long";
}
else if (len >5)
{
    for (int i = 0; i < input2; i++)
    {
        sb.Append(input1.Substring(len-3, 3));
    }
    return sb.ToString();
}
return " ";
}
}
```

81-100

#### 81.a.)81.String Processing II

Given a string input input1 ,form another string with the given input string using the following rules.

Form the output string with only the last letter of each word of the given input sentence in capital letters separated by \$. Dont store \$ after the last letter in the output string.

Example 1:

Input1:This is a cat

Output1:\$\$\$\$A\$T

Example 2:

Input1:This7 is a cat

Output1: -1

Business Rules :

1. If the given input string contains any number, print -1.
2. If the input contains any special characters, print -2.
3. If there is only one word in input1, then print -3.

Create a class named UserProgramCode that has the following static method

```
public static string formWordwithLastLetters(string 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 a string.

Output consists of a string or '-1' or '-2' or '-3'.

Sample Input 1:

This is a cat

Sample Output 1:

S\$\$\$A\$T

Sample Input 2:

This7 is a cat

Sample Output 2:

-1

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```



```

namespace ConsoleApplication13
{
    class userprogramcode
    {
        public static string formWordwithLastLetters(string ip1)
        {
            string[] s= new string[ip1.Length];
            char[] s1 = new char[ip1.Length];
            string final="";
            s = ip1.Split(' ');
            int i = 0;
            if (s.Length == 1)
            {
                return "-3";
            }
            foreach (string x in s)
            {
                s1[i] = x.ElementAt(x.Length - 1);

                if (char.IsNumber(s1[i]))
                {
                    return "-1";
                }
                else if (!char.IsLetter(s1[i]))
                {
                    return "-2";
                }

                i++;
            }
            int j=0;
            while (i > 0)
            {

```

```

        final = final + char.ToUpper(s1[j])+"$";

        j++;

        i--;

    }

    final = final.Remove(final.Length - 1);

    return final;

}

}

class Program
{
    static void Main(string[] args)
    {
        String x,y;

        x = Console.ReadLine();

        y = userprogramcode.formWordwithLastLetters(x);

        Console.WriteLine(y);

    }

}

}

```

another ethod:

```

using System;

using System.Collections.Generic;

using System.Linq;

```

```
using System.Text;
```

```
namespace ConsoleApplication18
```

```
{
```

```
    class Userprogramcode
```

```
    {
```

```
        public static string formWordwithLastLetters(string ip1)
```

```
        {
```

```
            char[] c = ip1.ToCharArray();
```

```
            StringBuilder sb = new StringBuilder();
```

```
            string[] s = ip1.Split(' ');
```

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

```
            {
```

```
                if (char.IsDigit(c[i]))
```

```
                    return "-1";
```

```
                else if (!char.IsLetter(c[i]) && !char.IsDigit(c[i]) && c[i] != ' ')
```

```
                    return "-2";
```

```
            }
```

```
            if (s.Length == 1)
```

```
            {
```

```
                return "-3";
```

```
            }
```

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

```
            {
```

```

        int a=s[i].Length;
        string g = s[i].Substring(a - 1).ToUpper();
        sb.Append(g);
        sb.Append('$');
    }
    string z=sb.ToString();
    string h=z.Remove(z.Length-1);
    return h;
}

```

```

    }
}

```

---

### 81.b.)String Processing I

Given a string input input1, write a program to fetch the last n characters from input1 and repeat them after input1 the same number of times as given in the second integer input input2.

Business Rules :

1. If the input1 contains any number, print -1.
2. If the input1 contains any special characters, print -2.
3. If the input1 string contains less than input2 number of characters,then print -3.

Create a class named UserProgramCode that has the following static method

```
public static string getString(string 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 a string and the second line of the input consists of an integer.

Refer sample output for formatting specifications.

Sample Input 1:

Cognizant

3

Sample Output 1:

Cognizantantantant

Sample Input 2:

Teach123er

4

Sample Output 2:

-1

using System;

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace ConsoleApplication13
```

```
{
```

```
    class userprogramcode
```

```
    {
```

```
        public static string getString(string ip1, int ip2)
```

```
        {
```

```
            string s,final=ip1;
```

```
            char s1;
```

```
            for (int i = 0; i < ip1.Length; i++)
```

```
            {
```

```
                s1 = ip1[i];
```

```
                if (char.IsNumber(s1))
```

```
                    return "-1";
```

```
                else if (!char.IsLetter(s1))
```

```
                    return "-2";
```

```
            }
```

```
            s = ip1.Substring(ip1.Length - (ip2));
```

```
            int j = ip2;
```

```
            while (j > 0)
```

```
            {
```

```
                final = final + s;
```

```

        j--;
    }
    return final;
}
}
class Program
{
    static void Main(string[] args)
    {
        String x,y;
        int k;
        x = Console.ReadLine();
        k = Convert.ToInt32(Console.ReadLine());
        y = userprogramcode.getString(x,k);
        Console.WriteLine(y);

    }
}
}
-----
-----

```

82.)82.Gyrating Numbers

Write a program to find whether every integer in a given input integer array is in Gyrating form.

Note: Gyrating numbers are numbers whose digits increase and decrease in a continuous repetitive cycle. Every integer of each element should increase or decrease in a continuous sequence.

Business rule:

- 1) Print 1 if the given input integer array is in Gyrating sequence.
- 2) Print -1 if the given input integer array is not in Gyrating sequence.
- 3) Print -2 if the given input integer array consists of a negative number.

Example:1

Input:

4

12

321

235

532

Output:

1

Example:2

Input:

4

75

12

531

45

Output:

1

Create a class named UserProgramCode that has the following static method



```
public static int gyRating(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 in the input correspond to the elements in the array.

Output is an integer. Refer business rules.

Sample Input :

```
4
12
321
235
532
```

Sample Output :

```
1
```

```
class UserProgramCode
{
    public int check(int[] a,int len)
    {
        int r,cmax=0,cmin=0,count=0,i=0,rem=0,j=0;
        StringBuilder sb = new StringBuilder();
```

```

foreach (int val in a)
{
    i = 0; cmax = 0; cmin = 0;

    int v = val;

    if (val < 0)
        return -2;

    else
    {
        while (v > 0)
        {
            ++j;

            r = v % 10;

            if (i == 0)
            {
                rem = r;

                ++i;
            }

            if (r == rem)
            {
                ++cmin;

                ++cmax;
            }

            else if (r < rem)
            {
                ++cmin;

                rem = r;
            }

            else if (r > rem)
            {
                ++cmax;
            }
        }
    }
}

```

```

        rem = r;
    }
    v = v / 10;
}
if (cmin == 1)
    sb.Append("m");
if (cmax == 1)
    sb.Append("M");
if (cmin == 1 || cmax == 1 )
{
    ++count;
}
else
    return -1;
}

}

if (count == len)
{
    //Console.WriteLine("true");
    int counter = 0;
    char odd = sb[1];
    char even = sb[0];
    if (!odd.Equals(even))
    {
        for (int ii = 0; ii < sb.Length; ii++)
        {
            if (ii % 2 == 0)
            {
                //Console.WriteLine("inside even pos");
            }
        }
    }
}

```

```
        if (sb[ii].Equals(even))
            counter++;
        else
            return -1;

    }

    else
    {
        //Console.WriteLine("inside odd pos");
        if (sb[ii].Equals(odd))
            counter++;
        else
            return -1;
    }

}

if (counter == count)
    return 1;

}

else
    return -1;

}

return 0;

}

}
```

---

### 83.83.String Array Sorting

Given a string array, write a function to remove the duplicate values from a String Array, sort the strings in ascending and display the string array.

The values 'AA' and 'aa' are NOT the same elements or duplicates. The case sensitive check should be implemented. While sorting, words starting with upper case letters should be considered first.

Business rules:

- 1) Print 'Invalid String' when the given input integer array consists of any special character or numbers.
- 2) All the elements in the array should be of same length. If not, then print 'Invalid String'.

Create a class named UserProgramCode that has the following static method

```
public static string[] orderStringElements(string[] 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 string array.

The next 'n' lines in the input correspond to the elements in the string array.

Output is a string array. Refer sample output and business rules

Sample Input 1:

6

AAA

BBB

AAA

AAA

CCC

CCC

Sample Output 1:

AAA

BBB

CCC

Sample Input 2:

7

AAA

BBB

aaa

AAA

Abc

A

b

Sample Output 2:

Invalid String

```
//program.cs
```

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;

namespace trial
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            String[] ar = new String[n];
            for (int i = 0; i < n; i++)
            {
                ar[i] = Console.ReadLine();
            }
            String[] ret = UserProgramCode.orderStringElements(ar);
            foreach (String a in ret)
            {
                Console.WriteLine(a);
            }
        }
    }
}
```

```
//UserProgramCode.cs

using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Text;
using System.Text.RegularExpressions;
namespace trial
{
    class UserProgramCode
    {
        public static string[] orderStringElements(string[] ar)
        {
            int len = ar[0].Length;
            String pat = @"^[a-zA-Z]{" + len + "}$";
            // Regex reg = new Regex(@"^[a-zA-Z]+$");
            Regex reg1 = new Regex(pat);
            String[] res = new String[1];
            StringBuilder sb = new StringBuilder();
            int n = ar.Length;
            foreach (String aa in ar)
            {
                if (!reg1.IsMatch(aa))
                {
                    res[0] = "Invalid String";
                    return res;
                }
            }
            for (int i = 0; i < n; i++)
            {
                String a = ar[i];

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

```



```

if (a.Equals(ar[j]) && !a.Equals(""))
{
    ar[j] = "-1";
}
}
if (!a.Equals("-1"))
{
    if (i > 0)
        sb.AppendLine();
    sb.Append(a);
}
}
String[] array = sb.ToString().Split('\n');

Array.Sort(array, StringComparer.Ordinal);
return array;

}
}
}

```

-----

-----

#### 84.)84.Interchange Characters

Write a program that accepts a string input and interchanges the first and last characters. Case sensitivity should be checked.

Business rules:

- 1) Print 'Invalid String' when the given input string consists of any special characters or numbers.
- 2) Print 'No Change' when the first and last characters of the input string is same and of the same case.

Example 1:

Input: Execute

Output: executE

Example 2:

Input: BoB

Output: No Change

Create a class named UserProgramCode that has the following static method

public static string interchangeFirstLast(string input1)

Create a class named Program that accepts the input and calls the static method present in the UserProgramCode.

Input and Output Format:

Input consists of a string.

Output is a string. Refer sample output and business rules

Sample Input 1:

Execute

Sample Output 1:

executE

Sample Input 2:

BoB

Sample Output 2:

No Change

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

namespace ConsoleApplication13
{
    class userprogramcode
    {
        public static string getString(string ip1)
        {
            string[] final = new string[ip1.Length];
            for (int j = 0; j < ip1.Length; j++)
                final[j] = ip1[j].ToString();
            string t1,ans="";
            for (int j = 0; j < ip1.Length; j++)
                if (!char.IsLetter(ip1[j]))
                    return "Invalid String";
            if (final[0] != final[ip1.Length - 1])
            {
                t1 = final[0];
                final[0] = final[ip1.Length - 1];
                final[ip1.Length - 1] = t1;
                for (int i = 0; i < ip1.Length; i++)
                    ans=ans+final[i];
            }
            else
```

```

        ans = "No Change";

        return ans;
    }
}

class Program
{
    static void Main(string[] args)
    {
        String x, y;

        x = Console.ReadLine();

        y = userprogramcode.getString(x);
        Console.WriteLine(y);

    }
}
}

-----
-----

```

85.)85.MaxMin Sum

Write a program that accepts 3 integer inputs and finds the sum of maximum and minimum.

Business Rules :

- 1) If any/ or all of the input value is negative then print -1.
- 2) If any two or all the values in the Input are same then print -2.

Example 1:

Input1: 25

Input2: 2

Input3: 95

Output : 97 (Min 2 + Max 95)

Example 2:

Input1: -15

Input2: 49

Input3: 5

Output : -1

Create a class named UserProgramCode that has the following static method

```
public static int sumMaxMin(int input1, int input2, int input3)
```

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 3 integers.

Output is an integer. Refer sample output and business rules

Sample Input 1:

25

2

95

Sample Output 1:

97

Sample Input 2:

-15

49

5

Sample Output 2:

-1

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace ConsoleApplication13
```

```
{
```

```
    class userprogramcode
```

```
    {
```

```
        public static int sumMaxMin(int ip1, int ip2, int ip3)
```

```
        {
```

```
            int ans,a,b;
```

```
            int[] t1 = new int[3];
```

```
            t1[0] = ip1;
```

```
            t1[1] = ip2;
```

```

        t1[2] = ip3;
        for (int i = 0; i < 3; i++)
            if (t1[i] < 0)
                return -1;
        for (int i = 0; i < 2; i++)
        {
            for (int j = i + 1; j < 3; j++)
            {
                if (t1[i] == t1[j])
                    return -2;
            }
        }
        a = t1.Max();
        b = t1.Min();
        ans = a + b;
        return ans;
    }
}

class Program
{
    static void Main(string[] args)
    {

        int x,y,z,k;
        x = Convert.ToInt32(Console.ReadLine());
        y = Convert.ToInt32(Console.ReadLine());
        z = Convert.ToInt32(Console.ReadLine());
        // k = Convert.ToInt32(Console.ReadLine());
        k =userprogramcode.sumMaxMin(x,y,z);
        Console.WriteLine(k);
    }
}

```

```
}  
}  
}
```

---

#### 86)86.Employee Designation

Given an input1 string array in the format {Employee1, Designation, Employee2, Designation, Employee3, Designation, and so on... } and a string input2, write a program to fetch the employee names from input1 based on input2 (designation) value and assign it in an output array and print the array. Case sensitivity can be ignored.

Business rule:

- 1) If input1 or input2 contains any special characters, then print 'Invalid Input'
- 2) If input1 does not contain the designation in input2, then print 'No employee for ABC designation' where ABC is the Input2 value.
- 3) If all the employees belong to the same designation, then print 'All employees belong to same ABC designation' where ABC is the Input2 value.

Example 1:

input1:

Ram

Manager

Ganesh

Developer



Srijith

Developer

input2:

Developer

output :

Ganesh

Srijith

Example 2:

Input 1:

Manish

BiDeveloper

Babu

Manager

Rohit

Associate

Input 2:

System Analyst

Output1:

No employee for System Analyst designation

Create a class named UserProgramCode that has the following static method

```
public static String[] getEmployee(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:

The first line of the input consists of an integer, n that corresponds to the number of elements in the string array.

The next 'n' lines of input consists of strings that correspond to elements in the string array.

The next line of the input consists of a string that corresponds to the Designation.

Refer business rules and sample output for output format.

Sample Input 1:

6

Ram

Manager

Ganesh

Developer

Srijith

Developer

Developer

Sample Output 1:

Ganesh

Srijith

Sample Input 2:

6

Manish

BiDeveloper

Babu

Manager

Rohit

Associate

System Analyst

Sample Output 2:

No employee for System Analyst designation

```

using System;
using System.Text.RegularExpressions;
namespace code1
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            Regex reg = new Regex(@"([A-Za-z])$");
            n = int.Parse(Console.ReadLine());
            String[] input1 = new String[n];
            String input2;

            String[] output;
            for (int i = 0; i < n; i++)
            {
                input1[i] = Console.ReadLine();
                if (!reg.IsMatch(input1[i]))
                {
                    Console.WriteLine("Invalid Input"); return;
                }
            }

            input2 = Console.ReadLine();

```

```
using System;
```

```

public class UserMainCode {
    public static int j;

    public static string[] getEmployee(string[] input1, string input2){

        int n = input1.Length;
        j = 0;
        String[] temp=new String[n];

        for (int i = 0; i <n;i=i+2)
        {
            if (input1[i + 1].Equals(input2))
            {
                temp[j] = input1[i];
                j++;
            }
        }
        return temp;
    }
}

```

---



---

87)87.Donations

Given 2 inputs,string array input1 and integer input2.The usercodes,locations and donations are appended as one element and stored in input1 in the following format,

ABCDEFGHI- here the ABC represents the usercode ,DEF represents the location and GHI represents the donation amount.

Write a program to find the total amount donated by the users who have the same location code given in input2 integer value.

Business rule:

- 1) If the string array contains any duplicates, then print -1.
- 2) If the string array contains any special characters, then print -2.

Create a class named UserProgramCode that has the following static method

```
public static int getDonation(string[] 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, n that corresponds to the number of elements in the string array.

The next 'n' lines of input consists of strings that correspond to elements in the string array.

The next line of the input consists of an integer that corresponds to the location code.

Refer business rules and sample output for output format.

Sample Input 1 :

4

123111241

124222456

145111505

124553567

111

Sample Output 1 :

746

Sample Input 2 :

4

123111241

124222456

124222456

124553567

111

Sample Output 2 :

-1

```
using System;
using System.Text.RegularExpressions;
namespace code1
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            Regex reg = new Regex(@"([A-Za-z0-9])$");
            n = int.Parse(Console.ReadLine());
```

```

String[] input1 = new String[n];
int input2;
int output;
for (int i = 0; i < n; i++)
{
    input1[i] = Console.ReadLine();
    if (!reg.IsMatch(input1[i]))
    {
        Console.WriteLine("-2"); return;
    }

    }

    for (int i = 0; i <n; i++)
    {
        for (int j = i+1; j < n; j++)
        {
            if(input1[i].Equals(input1[j]))
            { Console.WriteLine("-1");}
        }
    }

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

    output = UserMainCode.getDonation(input1, input2);
    Console.WriteLine(output);
}
}

}

```



---

```
using System;
```

```
public class UserMainCode
```

```
{
```

```
    public static int getDonation(string[] input1, int input2)
```

```
{
```

```
    String temp;
```

```
    int n=input1.Length;
```

```
    int output=0,don;;
```

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

```
{
```

```
    temp = input1[i].Substring(3, 3);
```

```
    if(int.Parse(temp)==input2){
```

```
        don=int.Parse(input1[i].Substring(6,3));
```

```
        output += don;
```

```
    }
```

```
}
```

```
    return output;
```

```
    }
```

```
}
```

-----  
-----  
  
//pg88

## 88. 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");

}

}

```

-----  
-----

#### 89) 89. Postal Tariff

Jack who stays at Delhi sends new year greetings by post to his friends within India.

He wants to know the total postal charges he needs to pay for sending the greetings to his friends.

There are two types of postal delivery. Normal Post(NP) and Speedy Post (SP).

The tariff rates for NP are as follows

- A. Postal Cost from Delhi to Bhopal(BP) is Rs 100
- B. Postal Cost from Delhi to Chennai(CH) is Rs 450
- C. Postal Cost from Delhi to Orissa(OS) is Rs 200

For Speedy Post additional 30% of normal Post tariff is charged.

The locations and the type of post Jack wants to send are given in the input array where each element of the array is in the format XXYY-where XX represents the location code and YY represents the type of postal delivery done.

Write a program to calculate the total cost Jack paid to send the greetings to his friends. Print the output in the following format.

Jacks spend Rs ZZZZ to send the greetings

where ZZZZ is the total charges calculated. Ignore case sensitivity of input strings.

Business rules:

1. If any of the location codes are other than BP,CH or OS,then print "Invalid location Code" .
2. If any of the postal delivery code is other than NP or SP, then print "Invalid Postal Delivery".

Create a class named UserProgramCode that has the following static method

```
public static void getPostalTariff(string[] 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 string array.

The next 'n' lines of input consists of strings that correspond to elements in the string array.

Refer business rules and sample output for output format.

Always display the total charges to be paid as an int.

Sample Input 1:

3

BPSP

CHNP

BPNP

Sample Output 1:

Jack spends Rs 680 to send the greetings

Sample Input 2:

3

BPSP

CHSP

PPNP

Sample Output 2:

Invalid location Code

```
using System;
using System.Text.RegularExpressions;
namespace code1
{
    class Program
    {
        static void Main(String[] args)
        {
            int n;
            n = int.Parse(Console.ReadLine());
            String[] input1=new String[n];

            for (int i = 0; i < n; i++)
            {
                input1[i] = Console.ReadLine();
```



```
}  
    UserMainCode.getPostalTariff(input1);
```

```
    }
```

```
}
```

```
}
```

---

```
using System;
```

```
public class UserMainCode
```

```
{
```

```
    public static void getPostalTariff(string[] input1)
```

```
{
```

```
    int length = input1.Length;
```

```
    double amount = 0;
```

```
    for (int i = 0; i < length; i++)
```

```

{

    if (input1[i].Substring(2, 2) == "SP")
    {
        if (input1[i].Substring(0, 2) == "BP")
            amount += (100*1.3);
        else if (input1[i].Substring(0, 2) == "CH")
            amount += (450 * 1.3);
        else if (input1[i].Substring(0, 2) == "OS")
            amount += (200 * 1.3);
        else
            { Console.WriteLine("Invalid location Code"); return; }

    }

    else if (input1[i].Substring(2, 2) == "NP")
    {
        if (input1[i].Substring(0, 2) == "BP")
            amount += (100);
        else if (input1[i].Substring(0, 2) == "CH")
            amount += (450);
        else if (input1[i].Substring(0, 2) == "OS")
            amount += (200);
        else
            { Console.WriteLine("Invalid location Code");return;}
    }

    else

```

```

    { Console.WriteLine("Invalid Postal Delivery"); return;}
}

Console.WriteLine("Jack spends Rs "+amount+" to send the greetings");
}
}

```

-----  
-----

#### 90) 90.Travel Agency

A travel agency has set standard tariffs for their pick up - drop services in a particular route. The route covers A,B,C,D locations one after the other.

A. Tariff for the travel from Location A to Location B is 10 units/Km

B. Tariff for the travel from Location B to Location C is 20 units/Km

C. Tariff for the travel from Location C to Location D is 40 units/Km

Return journey service is also provided.

The starting point, destination point and the Time of travel ( Normal - N, Untime - U) covered by a vehicle in a day are given as input1 in the format

{XYZ...} - here X represents Start point , Y represents the destination point and Z represents the Time of travel.

For untime travel, 20% additional charges are applicable on actual tariff for that route.

Write a program to calculate the total tariff collected by that vehicle for the day given and print the output in the following format,

The car has taken A trips and has collected total amount of C rupees.

-Here A refers to the total number of services provided per day and C refers to the total amount from all the travels.

Business rules:

- 1.If start point or destination points are invalid (other than A,B,C,D), print 'Invalid Location'.
- 2.If Time of travel is not either N or U , print 'Invalid Time of Travel'.

Create a class named UserProgramCode that has the following static method

```
public static int getTariffAmount(string[] 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 string array.

The next 'n' lines of input consists of strings that correspond to elements in the string array.

Refer business rules and sample output for output format.

Always display the tariff to be paid as an int.

Sample Input 1 :

4

ACN

DAU

ADN

DCU

Sample Output 1 :

The car has taken 4 trips and has collected total amount of 232 rupees

Sample Input 2 :

4

ACN

FAU

ADN

DCU

Sample Output 2 :

Invalid Location

```
using System;
using System.Text.RegularExpressions;
namespace code1
{
    class Program
    {

        static void Main(String[] args)
        {
            int n, amount;
            n = int.Parse(Console.ReadLine());
            String[] input1=new String[n];

            for (int i = 0; i < n; i++)
            {
                input1[i] = Console.ReadLine();
            }

            amount=UserMainCode.getTariffAmount(input1);
```

```
        if(amount!=-1&& amount!=-2)

            Console.WriteLine("The car has taken "+n+" trips and has collected total amount of " + amount + "
            rupees");

    }

}

}
```

```
using System;
```

```
public class UserMainCode
{
```

```
    public static int getTariffAmount(string[] input1)
    {
        int length = input1.Length;
        double amount = 0;
        for (int i = 0; i <length;i++)
        {
            if (input1[i][2] == 'N')
            {
                if (input1[i][0] == 'A')
                {
```

```
    if (input1[i][1] == 'B')
        amount += 10;
    else if (input1[i][1] == 'C')
        amount += 30;
    else if (input1[i][1] == 'D')
        amount += 70;
}
else if (input1[i][0] == 'B')
{
    if (input1[i][1] == 'A')
        amount += 10;
    else if (input1[i][1] == 'C')
        amount += 20;
    else if (input1[i][1] == 'D')
        amount += 60;
    else
    {
        Console.WriteLine("Invalid Location"); return -1;
    }
}
else if (input1[i][0] == 'C')
{
    if (input1[i][1] == 'A')
        amount += 30;
    else if (input1[i][1] == 'B')
        amount += 20;
    else if (input1[i][1] == 'D')
        amount += 40;
    else
    {
```

```
    Console.WriteLine("Invalid Location"); return -1;
}
}
else if (input1[i][0] == 'D')
{
    if (input1[i][1] == 'A')
        amount += 70;
    else if (input1[i][1] == 'B')
        amount += 60;
    else if (input1[i][1] == 'C')
        amount += 40;
    else
    {
        Console.WriteLine("Invalid Location"); return -1;
    }
}
else
{
    Console.WriteLine("Invalid Location"); return -1;
}
}
else if (input1[i][2] == 'U')
{
    if (input1[i][0] == 'A')
    {
        if (input1[i][1] == 'B')
            amount += 10 * 1.2;
        else if (input1[i][1] == 'C')
            amount += 30 * 1.2;
        else if (input1[i][1] == 'C')
```



```
    amount += 70 * 1.2;
}
else if (input1[i][0] == 'B')
{
    if (input1[i][1] == 'A')
        amount += 10 * 1.2;
    else if (input1[i][1] == 'C')
        amount += 20 * 1.2;
    else if (input1[i][1] == 'D')
        amount += 60 * 1.2;
    else
    {
        Console.WriteLine("Invalid Location"); return -1;
    }
}
else if (input1[i][0] == 'C')
{
    if (input1[i][1] == 'A')
        amount += 30 * 1.2;
    else if (input1[i][1] == 'B')
        amount += 20 * 1.2;
    else if (input1[i][1] == 'D')
        amount += 40 * 1.2;
    else
    {
        Console.WriteLine("Invalid Location"); return -1;
    }
}
else if (input1[i][0] == 'D')
{
```

```

    if (input1[i][1] == 'A')
        amount += 70 * 1.2;
    else if (input1[i][1] == 'B')
        amount += 60 * 1.2;
    else if (input1[i][1] == 'C')
        amount += 40 * 1.2;
    else
    {
        Console.WriteLine("Invalid Location"); return -1;
    }
}

else
{
    Console.WriteLine("Invalid Location"); return -1;
}

}

else
{ Console.WriteLine("Invalid Time of Travel"); return -2; }
}

return (int)amount;
}
}

```

---

91.91.Longest Palindrome

Given an input string input1, write a program to find the length of the longest substring which is a palindrome. Palindrome is a word, phrase, or sequence that reads the same backwards as forwards e.g. madam

Ignore case sensitivity for the input strings.

Business Rule:

- 1) If the input string contains any number, then print -1.
- 2) If the input string contains any special characters, then print -2.
- 3) If the input string does not contain a string palindrome, then print -3. Please note that a single character is not considered to be a palindrome.

Create a class named UserProgramCode that has the following static method

```
public static int longestPalindrome(string 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 a string.

Output is an integer.

Refer business rules and sample output for output format.

Sample Input 1:

seaesstringnirts

Sample Output 1:

11

Sample Input 2:

sea34esstringnirts

Sample Output 2 :

-1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Collections;
namespace ConsoleApplication23
{
    class UserProgramCode
    {
        public static int longestPalindrome(string s1)
        {
            string s = s1;
            s = Regex.Replace(s, @"\s+", " ");
            string[] s2 = s.Split(' ');
            char[] a2=new char[100];

            for(int i=0;i<s2.Length;i++)
            {
                for(int j=0;j<s2[i].Length;j++)
                {
                    if(char.IsDigit(s2[i][j]))
                    {
                        return -1;
                    }
                }
            }
        }
    }
}
```

```

    }
}
for (int i = 0; i < s2.Length; i++)
{
    for (int j = 0; j < s2[i].Length; j++)
    {
        if (!char.IsLetter(s2[i][j]))
        {
            return -2;
        }
    }
}
}

```

```

for (int i = 0; i < s2.Length; i++)
{
    int flag = 0;
    string r=s2[i];
    char[] a1=r.ToCharArray();
    int k = a1.Length;
    for (int i1 = 0; i < k; i1++)
    {

        a2[i1] = a1[k-1];

        k--;
    }
}

```

```
for (int j = 0; j < s2[i].Length; j++)  
{  
  
    if (s2[i][j] == a2[j])  
    {  
        flag = 1;  
    }  
    else  
    {  
        break;  
    }  
  
}  
if (flag == 1)  
{  
    return flag;  
}  
  
}  
  
return 0;  
  
}
```

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

```

{

    string a;
    a=Console.ReadLine();
    int flag;
    flag =longestPalindrome(a);
    Console.WriteLine(flag);
}
}
}

```

-----  
-----

92.)

92.Sum Largest Numbers In Range

Given an array of integer as input1 which falls under the range 1-100, write a program to find the largest numbers from input1 which would fall in the given range 1-10 , 11-20, 21-30, 31-40, ..... till 91-100. Now find their sum and print the sum.

Business Rules:

1. If the given input array contains any negative number then print -1.
2. If any element is equal to zero or greater than 100 then print -2.
3. In case the array of integer satisfies both business rule 1 as well as 2 then print -3.

4. In case of duplicate numbers eliminate the duplicate number and follow all other steps for calculation of the largest number.

Create a class named UserProgramCode that has the following static method

```
public static int largestNumber(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 size of the array.

The next 'n' lines of input consist of integers that correspond to the elements in the array.

Output is an integer.

Refer business rules and sample output for output format.

Sample Input 1:

```
7
13
18
26
34
58
65
54
```

Sample Output 1 :

```
201
```

Sample Input 2 :

```
5
-1
```



19

15

18

101

Sample Output 2 :

-3

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace sumlargest92
```

```
{
```

```
    class Program
```

```
    {
```

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

```
        {
```

```
            int i, n, li=0,c=0,c1=0,c2=0,c3=0,c4=0,c5=0,c6=0,c7=0,c8=0,c9=0;
```

```
            int s = 0;
```

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

```
            int[] a=new int[n];
```

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

```
            {
```

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

```

        if((a[i]==0)||a[i]>100))
        {
            Console.WriteLine(-2);
            System.Environment.Exit(1);
        }
        else if((a[i]<0)&&((a[i]==0)||a[i]>100)))
        {
            Console.WriteLine(-3);
            System.Environment.Exit(1);
        }
        else if(a[i]<0)
        {
            Console.WriteLine(-1);
            System.Environment.Exit(1
    );

        }
    }
}

for(i=0;i<n;i++)
{
    if (a[i] > 0 && a[i] <= 10)
    {
        if (c == 0)
        {
            s += a[i];
        }
        else
        {
            if (a[li] < a[i])
            {
                s = s - a[li];
            }
        }
    }
}

```

```
        s += a[i];  
    }  
}  
c++;  
li = i;  
}
```

```
else if (a[i] > 10 && a[i] <= 20)
```

```
{  
    if (c1 == 0)  
    {  
        li = 0;  
        s += a[i];  
    }  
    else  
    {  
        if (a[li] < a[i])  
        {  
            s = s - a[li];  
            s += a[i];  
        }  
    }  
    c1++;  
    li = i;  
}
```

```
else if (a[i] > 20 && a[i] <= 30)
```

```
{  
    if (c2 == 0)  
    {  
        li = 0;
```

```

        s += a[i];
    }
else
{
    if (a[li] < a[i])
    {
        s = s - a[li];
        s += a[i];
    }
}
c2++;
li = i;
}
else if (a[i] > 30 && a[i] <= 40)
{
    if (c3 == 0)
    {
        li = 0;
        s += a[i];
    }
else
{
    if (a[li] < a[i])
    {
        s = s - a[li];
        s += a[i];
    }
}
c3++;
li = i;

```

```

}
else if (a[i] > 40 && a[i] <= 50)
{
    if (c4 == 0)
    {
        li = 0;

        s += a[i];
    }
    else
    {
        if (a[li] < a[i])
        {
            s = s - a[li];

            s += a[i];
        }
    }
    c4++;
    li = i;
}
else if (a[i] > 50 && a[i] <= 60)
{
    if (c5 == 0)
    {
        li = 0;

        s += a[i];
    }
    else
    {
        if (a[li] < a[i])
        {

```

```

        s = s - a[li];

        s += a[i];

    }

}

c5++;

li = i;

}

else if (a[i] > 60 && a[i] <= 70)

{

    if (c6 == 0)

    {

        li = 0;

        s += a[i];

    }

    else

    {

        if (a[li] < a[i])

        {

            s = s - a[li];

            s += a[i];

        }

    }

    c6++;

    li = i;

}

else if (a[i] > 70 && a[i] <= 80)

{

    if (c7 == 0)

    {

        li = 0;


```

```

        s += a[i];
    }
else
{
    if (a[li] < a[i])
    {
        s = s - a[li];
        s += a[i];
    }
}
c7++;
li = i;
}
else if (a[i] > 80 && a[i] <= 90)
{
    if (c8 == 0)
    {
        li = 0;
        s += a[i];
    }
else
{
    if (a[li] < a[i])
    {
        s = s - a[li];
        s += a[i];
    }
}
c8++;
li = i;

```

```
    }  
    else if (a[i] > 90 && a[i] <= 100)  
    {  
        if (c9 == 0)  
        {  
            li = 0;  
            s += a[i];  
        }  
        else  
        {  
            if (a[li] < a[i])  
            {  
                s = s - a[li];  
                s += a[i];  
            }  
        }  
        c9++;  
        li = i;  
    }  
  
}
```

```
Console.WriteLine("sum="+s);
```



```
    }  
  }  
}
```

-----  
-----

93.Next Consonant or Vowel

93.Next Consonant or Vowel

Given an input String, write a program to replace all the vowels of the given string with the next consonant and replace all consonants with the next available vowel.

Business Rule:

1. If the input string contains any number or any special characters, print 'Invalid input'.
2. The input is case sensitive. Please ensure that each character in the output has exactly the same case as the input string.

Create a class named UserProgramCode that has the following static method

```
public static string nextString(String 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 a string.

Output consists of a string. Refer business rules and sample output for the format.

Sample Input 1 :

zebRa

Sample Output 1 :

afeUb

Sample Input 2 :

cat@rat/123

Sample Output 2 :

Invalid input

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Reflection;
```

```
namespace ConsoleApplication2  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            string str = Console.ReadLine();  
            Console.WriteLine(UserProgramCode.nextString(str));  
        }  
    }  
}
```

```
    }  
}  
  
}
```

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Text.RegularExpressions;  
using System.Collections;
```

```
namespace ConsoleApplication2  
{  
    class UserProgramCode  
    {  
        public static string nextString(string str)  
        {
```

```
            ArrayList vowel_small = new ArrayList();  
            ArrayList vowel_caps = new ArrayList();
```

```
vowel_small.Add('a');
vowel_small.Add('e');
vowel_small.Add('i');
vowel_small.Add('o');
vowel_small.Add('u');
vowel_caps.Add('A');
vowel_caps.Add('E');
vowel_caps.Add('I');
vowel_caps.Add('O');
vowel_caps.Add('U');
char[] inp = str.ToCharArray();
char[] out1 = new char[str.Length];

for (int i = 0; i < str.Length;i++ )
{
    if (vowel_caps.Contains(inp[i]) || vowel_small.Contains(inp[i]))
    {
        char ch = (char)((int)inp[i] + 1);
        out1[i] = ch;

        }
    else if (inp[i] == 90 || inp[i] == 122)
    {
        if (inp[i] == 90)
        out1[i]='A';
        else
        out1[i] = 'a';
    }
}
```

```
else
{
if (inp[i] >= 65 && inp[i] <= 90)
{
if (inp[i] > 85)
out1[i] = 'A';
else
{
foreach (char che in vowel_caps)
{
if (che > inp[i])
{
out1[i] = che;
break;
}
}
}
}
else
{
if (inp[i] > 117)
out1[i] = 'a';
else
{
foreach (char che in vowel_small)
{
if (che > inp[i])
{
out1[i] = che;
break;
}
```

```
        }  
    }  
}  
}  
}  
}  
}  
string output=null;  
foreach (char c in out1)  
{  
  
    output += c.ToString();  
}  
return output;  
  
}  
}  
}
```

-----  
-----

94.Power of 2

94.Power of 2

Write a program to check whether an integer number is a power of 2 or not. If it is a power of 2 print the power else print -1 .

Business Rule:

1. If the given input integer is a negative number/not a power of 2, print -1.

Create a class named UserProgramCode that has the following static method

public static int twoPower(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 consists of an integer. Refer business rules and sample output for the format.

Sample Input 1 :

1024

Sample Output 1 :

10

Sample Input 2 :

6

Sample Output 2 :

-1

using System;

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Collections;
namespace ConsoleApplication23
{
    class UserProgramCode
    {
        public static int twoPower(int input1)
        {
            int i1 = input1;
            int n=2;
            int i = 1;
            int sum=1;
            int val=0;
            ArrayList al = new ArrayList();
            while (i < i1)
            {
                if ((sum * n) == i1)
                {
                    val = i;
                    break;
                }
                sum = sum * n;
                al.Add(sum);
                i++;
            }

            return val;
        }
    }
}
```



```

    }
    static void Main(string[] args)
    {
        int i;
        i = int.Parse(Console.ReadLine());
        int val1;
        val1 = twoPower(i);
        if (val1 > 0)
            Console.WriteLine(val1);
        else
            Console.WriteLine(-1);
    }
}

```

-----  
-----

95.String Equal Check

95.String Equal Check

Given two strings Input1 and Input2 and integer Input3, write a program to check if Nth character of Input1 traversing from first and Nth character of Input2 traversing from last are same irrespective of case where N is the Input3 value. Ignore case.

If both are same, then print "The character is x" where x is the Nth character

If both are not same, then print "The character x and y does not match" where x is the Nth character of Input1 starting from first and y is the Nth character of Input2 starting from last.

Business rule:

- 1) If the Input1 string contains any special characters or numbers, then print 'Invalid Input'
- 2) If the Input2 string contains any special characters or numbers, then print 'Invalid Input'
- 3) If the Input3 value is greater than the length of Input1 and/or Input2, then print 'Invalid Input'

Create a class named UserProgramCode that has the following static method  
`public static string stringEqualCheck(string input1, string input2, int input3)`

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 a string that corresponds to input1.

The second line of the input consists of a string that corresponds to input2.

The third line of the input consists of an integer that corresponds to input 3.

Output consists of a string. Refer business rules and sample output for the format.

Sample Input 1:

Battle

Final

2

Sample Output 1:

The character is a

Sample Input 2 :

Photograph

Anticipate

4

Sample Output 2:

The character t and p does not match

Sample Input 3 :

xerox

pretty

15

Sample Output 3 :

Invalid Input

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Collections;
namespace ConsoleApplication23
{
    class UserProgramCode
    {
        public static string stringEqualCheck(string input1, string input2, int input3)
        {
            string i1, i2;
            int i3;

            i1 = input1;
            char[] i11=i1.ToCharArray();
            i2 = input2;
            char[] i22=i2.ToCharArray();
```

```

i3 = input3;
foreach (char ch in i11)
{
    if (char.IsDigit(ch))
    {
        Console.WriteLine( "invalid input");
    }
    if (!char.IsLetter(ch))
    {
        Console.WriteLine("invalid input");
    }
}
foreach (char ch in i22)
{
    if (char.IsDigit(ch))
    {
        Console.WriteLine("invalid input");
    }
    if (!char.IsLetter(ch))
    {
        Console.WriteLine("invalid input");
    }
}
if(i3>i1.Length)
{
    Console.WriteLine("invalid input");
}
else if (i3 > i2.Length)
{
    Console.WriteLine("invalid input");
}

```

```
}
```

```
else if ((i11[i3 - 1]) == (i22[(i2.Length) - i3]))  
{  
    Console.WriteLine("the character is" + i11[i3 - 1]);  
}  
else  
{  
    Console.WriteLine("the character " + i11[i3 - 1] + "and" + i22[(i2.Length) - i3] + " does not  
match");  
}
```

```
    return "string";  
}  
static void Main(string[] args)  
{  
    string s1, s2;  
    int index;  
    s1 = Console.ReadLine();  
    s2 = Console.ReadLine();  
    index = int.Parse(Console.ReadLine());  
    string disp;  
    disp = stringEqualCheck(s1, s2, index);
```

```
    }  
  }  
}
```

96.

#### 96.3,4 Number System

Given a number system having numbers which is a combination of digits 3 and 4 only. First few numbers in the number system are: 3, 4, 33, 34, 43, 44, 333, 334, 343, 344, 433, 434, 443, 444, 3333, 3334, 3343, 3344, 3433, 3434, 3443, 3444, ... Find the nth number in the number system where n is an integer index given as input .

Business Rule:

1. If the input1 is less than 1, then print -1

Create a class named UserProgramCode that has the following static method

```
public static void findNumber(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 is an integer.

Output is an integer.

Sample Input 1 :

10

Sample Output1 :

344

Sample Input 2 :

-8

Sample Output 2 :

-1

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace ConsoleApplication96
```

```
{
```

```
    class Program
```

```
    {
```

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

```
        {
```

```
            int n;
```

```
            n = Convert.ToInt16(Console.ReadLine());
```

```
            UserProgramCode.findNumber(n);
```

```
        }
```

```
    }
```

```
    class UserProgramCode
```

```
    {
```

```
        public static void findNumber(int input1)
```

```

{

    int i = 0;
    int count = 0;
    if (input1 > 0)
    {
        while (1 == 1)
        {
            int x = ++i;
            int z = x;
            string k = Convert.ToString(x);
            int l = k.Length;
            int cc = 0;
            for (int j = 0; j < l; j++)
            {
                int a = x % 10;
                int b = x / 10;
                x = b;
                if (a == 3 || a == 4)
                {
                    cc++;
                }
                else
                {
                    break;
                }
            }
            if (cc == 1)
            {
                count++;
            }
            if (input1 == count)
            {
                Console.WriteLine(z);
            }
        }
    }
}

```



```

        break;
    }
}
else
    Console.WriteLine(-1);
}
}
}

```

=====

#### 97.97.Rearrange Case

Given a string input, write a program to form a new string provided with the the below limitations

1. Check for the alphabet which has maximum number of Upper case and lower case in the input string value.
2. Uppercase alphabets would be moved to the start of the Output string and lowercase alphabets should be moved to the end of the string.
3. Remaining other alphabets will remain the same in between the start and end of the output variable irrespective of the case.

Business rule:

- 1) If the Input string contains any special characters, then print 'Invalid Input'.
- 2) If the Input string does not contain Uppercase at all, then print 'Condition does not meet' .
- 3) If two or more alphabets has maximum upper and lower case, then print 'Re-arranging is not possible' .

Create a class named UserProgramCode that has the following static method

```
public static string rearrangeCase(string 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 a string.

Output consists of a string.

Refer business rules and sample output for the format.

Sample Input 1 :

CancelPolicy

Sample Output 1 :

CanelPoliycc

Sample Input 2 :

XYZbossxyz

Sample Output 2 :

Re-arranging is not possible

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Collections;
```

```
namespace ConsoleApplication97
```

```

{
class Program
{
    static void Main(string[] args)
    {
        string a, b;

        a = Console.ReadLine();
        b=UserProgramCode.rearrangeCase(a);
        Console.WriteLine(b);


    }
}
class UserProgramCode
{
    static char cf;

    public static string rearrangeCase(string input1)
    {
        int l = input1.Length;
        // string a=Convert.ToString();
        int c1 = 0,c3=0;


        StringBuilder a1 = new StringBuilder();
        StringBuilder a2 = new StringBuilder();
        StringBuilder a3 = new StringBuilder();


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

```

```
if(char.IsLower(input1[i]))
{
    c1++;
}
}
if (c1 == 1)
{
    return ("Condition does not meet");
}
else
{
```

```
for (int i = 0; i < l; i++)
{
    int c2 = 0;
    if (char.IsUpper(input1[i]))
    {
        for (int j = 0; j < l; j++)
        {
            if (input1[j] == char.ToLower(input1[i]))
                c2++;
        }
    }
    if (c2 > c3)
    {
        c3 = c2;
        cf = input1[i];
    }
    else if (c2 == c3)
```

```

        {
            return ("Re-arranging is not possible");
        }

    }

    for (int i = 0; i < l; i++)
    {
        if (input1[i] == char.ToUpper(cf))
        {
            a1.Append(cf);
        }
        else if (input1[i] == char.ToLower(cf))
        {
            a2.Append(char.ToLower(cf));
        }
        else
        {
            a3.Append(input1[i]);
        }
    }
}

return (a1.ToString() + "" + a3.ToString() + "" + a2.ToString());
}

}
}

```

```

=====
=====

```

98.98.Repeated Integers

Write code to pick all the repeated integers in a given integer array, sort them in ascending order and put them in the output list. Print the output list.

Include a class `UserProgramCode` with a static method `findRepeatedIntegers` which accepts the size of an integer array and an integer array. The return type is void. Print the repeated integers in sorted order if present. If there are no repeated numbers, then print "No repeated numbers". If there are negative numbers in the array, print "Array contains negative numbers".

Create a Class Program which would be used to accept Input array and call the static method present in `UserProgramCode`.

Input and Output Format:

Input consists of  $n+1$  integers. The first integer corresponds to  $n$ , the number of elements in the array. The next ' $n$ ' integers correspond to the elements in the

.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 30.

Sample Input 1:

4  
3  
3  
2  
10

Sample Output 1:

3

Sample Input 2:

4

3

1

2

10

Sample Output 2:

No repeated numbers

Sample Input 3:

4

3

-11

2

10

Sample Output 3:

Array contains negative numbers

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Collections;
```

```
namespace ConsoleApplication98
{
    class Program
    {
        static void Main(string[] args)
        {

            int l=1;
            int[] a = new int[30];
            for (int i = 0; i <= l; i++)
            {
                a[i] = Convert.ToInt16(Console.ReadLine());
                l = a[0];
            }
            UserProgramCode.findRepeatedIntegers(a);
        }
    }
}

class UserProgramCode
{
    public static void findRepeatedIntegers(int[] a)
    {
        ArrayList a1 = new ArrayList();
        int flag = 0;
        for (int i = 1; i <= a[0]; i++)
        {
            if (a[i] >= 0)
            {
                int c = 0;
                for (int j = 1; j <= a[0]; j++)
```



```

    {
        if (a[i] == a[j])
            c++;
    }
    if (c > 1)
    {

        if (!(a1.Contains(a[i])))
            a1.Add(a[i]);
    }
}
else
{
    flag = 1;
    break;
}
}
if (flag == 0)
{
    a1.Sort();
    int c1 = 0;
    foreach (int i in a1)
        c1++;
    if (c1 == 0)
        Console.WriteLine("No repeated numbers");
    else
    {
        foreach (int i in a1)
        {

```

```

        Console.WriteLine(i);
    }
}
else
    Console.WriteLine("Array contains negative numbers");
}
}
}

```

=====

99.

#### 99.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;

}

}

}

```

```

=====
=====

```

100.

100.Largest Digit

Write a program to find the Largest digit from given input integer. Print the largest digit. If the number is negative, print “Negative Number”.

Example:

Input1: 524

Output1: 5

Include a class UserProgramCode with a static method findLargestDigit which accepts an integer. The return type (integer) should return the largest digit. Return -1 if the number is negative.

Create a Class Program which would be used to accept an integer and call the static method present in UserProgramCode.

Input and Output Format:

Input consists of an integer.

Output consists of an integer (the largest digit), or a String “ Negative Number” if the input is negative.

Refer sample output for formatting specifications.

Sample Input 1:

524

Sample Output 1:

5

Sample Input 2:

-23

Sample Output 2:

Negative Number

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace ConsoleApplication100
```

```
{
```

```
    class Program
```

```
    {
```

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

```

{
    int a;
    a = int.Parse(Console.ReadLine());
    int c = UserProgramCode.findLargestDigit(a);
    if(c== -1)
        Console.WriteLine("Negative Number");
    else
        Console.WriteLine(c);
}
}
class UserProgramCode
{
    public static int findLargestDigit(int b)
    {
        if (b >= 0)
        {
            string a = b.ToString();
            int l = a.Length;
            int c = 0;
            for (int i = 0; i < l; i++)
            {
                int x = b / 10;
                int y = b % 10;
                b = x;
                if (y >= c)
                {
                    c = y;
                }
            }
            return c;
        }
    }
}

```



```
    }
    else
        return (-1);

    }
}
}

=====
=====
```

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

namespace permutation
{
    class Program
    {
        static void Main(string[] args)
        {
            string s = Console.ReadLine();
            List<string> op = Class1.perm(s);
            foreach (var l in op)
            {
                Console.WriteLine(l);
            }
        }
    }
}
```

```
    }  
    }  
}
```

## PERMUTATIONS WITH SORTING:

program.cs:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace permutation  
{  
    class Class1  
    {  
        public static List<string> perm(string s)  
        {  
            List<string> ans = recper("", s);  
            return ans;  
        }  
        static List<string> op = new List<string>();  
        public static List<string> recper(string a, string s)  
        {  
  
            if (string.IsNullOrEmpty(s))  
            {
```

```

        op.Add(a);
    }
    else
    {
        for (int i = 0; i < s.Length; i++)
        {
            string h = s.Substring(0, i) + s.Substring(i + 1);
            recper(a + s[i], h);
        }

    }
    return op;
}
}
}

```

-----

## CONVERT NUMBER TO WORDS:

```

class Program
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());
        string s=UserCode.ConvertNumbertoWords(n);
        Console.WriteLine(s);
    }
}

```

```
}
```

```
}
```

UserCode.cs:

```
class UserCode
{
    public static string ConvertNumbertToWords(int number)
    {
        if (number == 0)
            return "ZERO";
        if (number < 0)
            return "minus " + ConvertNumbertToWords(Math.Abs(number));
        string words = "";
        if ((number / 1000000) > 0)
        {
            words += ConvertNumbertToWords(number / 1000000) + " MILLION ";
            number %= 1000000;
        }
        if ((number / 1000) > 0)
        {
            words += ConvertNumbertToWords(number / 1000) + " THOUSAND ";
            number %= 1000;
        }
        if ((number / 100) > 0)
        {
            words += ConvertNumbertToWords(number / 100) + " HUNDRED ";
            number %= 100;
```

```

    }
    if (number > 0)
    {
        if (words != "")
            words += "AND ";

        var unitsMap = new[] { "ZERO", "ONE", "TWO", "THREE", "FOUR", "FIVE", "SIX", "SEVEN",
                                "EIGHT", "NINE", "TEN", "ELEVEN", "TWELVE", "THIRTEEN", "FOURTEEN", "FIFTEEN",
                                "SIXTEEN", "SEVENTEEN", "EIGHTEEN", "NINETEEN" };

        var tensMap = new[] { "ZERO", "TEN", "TWENTY", "THIRTY", "FORTY", "FIFTY", "SIXTY",
                                "SEVENTY", "EIGHTY", "NINETY" };

        if (number < 20)
            words += unitsMap[number];
        else
        {
            words += tensMap[number / 10];
            if ((number % 10) > 0)
                words += " " + unitsMap[number % 10];
        }
    }
    return words;
}
}

```

-----

LONGEST WORD STRING

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```
using System.Text;
```

```
namespace str15
```

```
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int n = int.Parse(Console.ReadLine());  
            string[] s = new string[n];  
  
            for (int i = 0; i < n; i++)  
            {  
                s[i] = Console.ReadLine();  
  
            }  
            char ch = Console.ReadKey().KeyChar;  
            Console.WriteLine( user.lg(s, n, ch));  
        }  
    }  
}
```

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace str15
```

```
{
```

```

class user
{
    public static string lg(string[] a, int sz,char fn)
    {

        string op=" ";
        int oplen=0;
        foreach (var item in a)
        {
            int f = 1;
            char[] c = item.ToCharArray();
            string g = fn.ToString();
            foreach (char i in c)
            {
                if ((!char.IsLetter(i)) && (!char.IsDigit(i)))
                    f = 2;
            }
            if (f == 1)
            {

                if (item.StartsWith(g))
                {
                    int len = item.Length;
                    if (oplen < len)
                    {
                        oplen = len;
                        op = item;
                    }
                }
                else
                {

```

```

        op = "no common";

    }

}

else if(f==2)
{
    op = "no alphabet ";
    return op;
    System.Environment.Exit(0);
}

}

return op;
}

}

}

```

---

REPEAT

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```

namespace elementrepeat
{

```



```

class Program
{
    static void Main(string[] args)
    {
        int n = int.Parse(Console.ReadLine());
        int[] a = new int[n];
        for (int i= 0; i <n ; i++)
        {
            a[i] = int.Parse(Console.ReadLine());
        }
        int[] ans = UserMainCode.rep(a, n);
        foreach (int item in ans)
        {
            Console.WriteLine(item);
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace elementrepeat
{
    class UserMainCode
    {
        public static int[] rep(int[] b,int s)
        {
            int[] op;

```

```
int k=0;

int[] temp=new int[s];
for (int i = 0; i < s; i++)
{
    for (int j = i+1; j < s; j++)
    {
        if (b[i] == b[j])
        {
            temp[k] = b[i];
            k++;
        }

    }

}

if (k == 0)
{
    op = new int[1];
    op[0] = -1;
}
else
{
    op = new int[k];
}

for (int i = 0; i < k; i++)
{
    op[i] = temp[i];
}
```

```
    Array.Sort(op);  
    return op;  
}
```

```
    }  
}
```

-----

class Program

```
{  
    static void Main(string[] args)  
    {  
        string s1=" ";  
        int n = int.Parse(Console.ReadLine());  
        string[] s = new string[n];  
        for (int i = 0; i < n; i++)  
        {  
            s[i] = Console.ReadLine();  
        }  
        char c = Console.ReadKey().KeyChar;  
        int max = s[0].Length;  
        for (int i = 0; i < n; i++)  
        {  
            if(s[i].StartsWith(c.ToString()))  
            {
```

```

        if (s[i].Length > max)
        {
            s1 = s[i];
        }
    }

    Console.WriteLine(s1);

}
}

```

---

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```

namespace ConsoleApplication2
{
    class Userprogramcode
    {
        public static int[] GetElements(int []a,int b1)
        {
            int e;
            e = a.Length;
            int[] c = new int[e];
            int k = 0;
            int flag=0;
            int[] p;

```

```
for (int i = 0; i < e; i++)
{
    if (a[i] > b1)
    {
        flag = 1;
        c[k] = a[i];
        k++;
    }
}

if (k == 0)
{
    k++;
    p = new int[k];
}
else
{
    p = new int[k];
}
if (flag == 0)
    p[0] = -1;
else
{
    for (int i = 0; i < k; i++)
    {
        p[i] = c[i];
    }
}
return p;
}
```

```
    }  
}
```

4.

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace ConsoleApplication2
```

```
{
```

```
    class Program
```

```
    {
```

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

```
        {
```

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

```
            int[] a = new int[n];
```

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

```
            {
```

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

```
            }
```

```
            int b = int.Parse(Console.ReadLine());
```

```
            int[] d= Userprogramcode.GetElements(a, b);
```

```
            if (d[0] == -1)
```

```
                Console.WriteLine("invalid input");
```

```
            else
```

```
            {
```

```
                foreach (int item in d)
```

```
    {  
        Console.WriteLine(item);  
    }  
}
```

-----

## IP ADDRESS

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace ipaddress  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            string s = Console.ReadLine();  
            Console.WriteLine(Userprgmclass.ip(s));  
        }  
    }  
}  
  
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;
```

```
using System.Net;
```

```
namespace ipaddress
```

```
{
```

```
    class Userprgmclass
```

```
    {
```

```
        public static string ip(string a)
```

```
        {
```

```
            IPAddress ip;
```

```
            string l = " ";
```

```
            bool b=IPAddress.TryParse(a, out ip);
```

```
            if (b)
```

```
            {
```

```
                l= "valid";
```

```
            }
```

```
            else
```

```
                l= "invalid";
```

```
            return l;
```

```
        }
```

```
    }
```

```
}
```