Triplets

Given inputs ,integer array input and an integer value 'k'. Write a program to find the three elements in the array whose sum is equal to the given input value 'k' and store the output in an array. Store the output elements in the same order as present in input array.

Business rule:
1) If any of the inputs is negative, then print -1.
2) If the input array does not contains the triplets whose sum is equal to integer k, then print -2.
3) If the input array contains any duplicates characters, then print -3.

Create a class named UserProgramCode that has the following static method
public static int[] findTriplets(int[] input1,int k)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.
The next 'n' lines of input consist of elements in the input array.
The next line of the input corresponds to the value of k.

Refer business rules and sample output for formatting specifications.

Sample Input 1 :
6
12
3
4
1
6
9
24

Sample Output 1 :
12
3
9

Sample Input 2 :
6
12
3

4
-1
6
9
16

Sample Output 2 :
-1

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;


namespace ConsoleApplication1

{

    class Program

    {

        static void Main(string[] args)

        {

            int num = Int32.Parse(Console.ReadLine());

            int[] a = new int[num];


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

            {

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

```
            }
            int val = int.Parse(Console.ReadLine());
            int[] r = UserProgramCode.findTriplets(a, val);
            foreach (int h in r)
            {
                Console.WriteLine(h);
            }
        }
    }
}
```

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static int[] findTriplets(int[] a, int val)
        {
            int num = a.Length;
            int l;
```

```
int[] b = new int[3];

l = 0;

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

{

    if (a[i] < 0)

    {

        b[0] = -1;

        return b;

    }

}


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

{

    for (int j = i + 1; j < num; j++)

    {

        for (int k = j + 1; k < num; k++)

        {

            if (a[i] + a[j] + a[k] == val)

            {

                b[l] = a[i];

                b[l + 1] = a[j];

                b[l + 2] = a[k];

                l = l + 3;

            }

            else if (a[i] == a[j])

            {
```

```
                        b[0] = -3;
                        return b;
                    }
                    else
                    {
                        b[0] = -2;
                        return b;
                    }

                }
            }

        }
        return b;
    }

}
}
```

Sum of Odd Even Positioned

Write a program to find whether the sum of digits at even indexes and sum of digits at odd indexes in the given number are equal.

Include a class **UserProgramCode** with a static method **sumOfOddEvenPositioned** that accepts an integer .The return type (integer) should return 1 if it a valid, else return 2.

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

Input and Output Format:

Input consists of an integer.

Output consists of a String("Valid" or "Not Valid").

Refer sample output for formatting specifications.

Sample Input 1:
1221

Sample Output 1:

Valid

Sample Input 2:

2000204

Sample Output 2:

Not Valid

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int num = Int32.Parse(Console.ReadLine());
            int r = UserProgramCode.findTriplets(num);
            if (r == 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 ConsoleApplication1
{
    class UserProgramCode
    {
        public static int findTriplets(int b)
        {
            int i = 0,even=0,odd=0,d;
            while (b > 0)
            {
                d = b % 10;
                if (i % 2 == 0)
                {
                    even += d;
                }
                else
                    odd += d;
                i++;
                b = b / 10;
```

```
            }
            if (even == odd)
            {
                return 1;
            }
            else
                return -1;
        }
    }
}
```

Check Batch Code

Write a program which will check if the given input string follows the below format and print the

output according to the conditions given below.

1. The format of the string should be 'AAABBCCXXX' where
a. AAA represents the location of the batch
CHN -- Chennai
CBE -- Coimbatore
KOC - Kochi
PUN - Pune
BGL - Bangalore
HYD - Hyderabad
KOL - Kolkata
Business rules:
THe characters 'AAA' should not be other than the above specified values(Only Capitals). If it is other than these characters, print -1.
b. BB and XXX in the format represents numerals between 0-9. BB Represents the year and XXX represents the batch code.If other than these are present print -2.
c.CC in the format should be only 'DN', if not print -3.
All the characters in the input string are in upper case. Please make sure you dont do a spell mistake in the output string.

Example 1:
Input : CHN13DN014
The output string should be in the following format.
DotNet batch 014 has joined in 2013 year and is at Chennai Location

Create a class named UserProgramCode that has the following static method
public static string checkBatch(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.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
CHN13DN014

Sample Output 1 :
DotNet batch 014 has joined in 2013 year and is at Chennai Location

Sample Input 2 :
PUN13DN004

Sample Output 2 :
DotNet batch 004 has joined in 2013 year and is at Pune Location

Sample Input 3 :
BGL14DN014

Sample Output 3 :
DotNet batch 014 has joined in 2014 year and is at Bangalore Location

```
 using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;


namespace dummy

{

   class Class1

   {

      public static string checkcode(string str)

      {

         int res=0;

         string res3 = string.Empty;

         string res2 = string.Empty;

         StringBuilder sb = new StringBuilder();

         string res1=string.Empty;

         string s1 = str.Substring(0, 3);
```

```
Regex r1 = new Regex(@"^(CHN|CBE|KOC|PUN|BGL|HYD|KOL)$");
bool rs1 = r1.IsMatch(s1);

if (rs1)
{
    if (s1 == "CHN")
        res1 = "Chennai";
    else if (s1 == "CBE")
        res1 = "Coimbatore";
    else if (s1 == "KOC")
        res1 = "Kochi";
    else if (s1 == "PUN")
        res1 = "Pune";
    else if (s1 == "BGL")
        res1 = "Banglore";
    else if (s1 == "HYD")
        res1 = "Hyderabad";
    else if (s1 == "KOL")
        res1 = "Kolkota";

}
else
{
```

```
    res = -1;
    return "-1";
}

string s2 = str.Substring(3, 2);
Regex r2 = new Regex(@"^([0-9]{2})$");
bool rs2 = r2.IsMatch(s2);
if (rs2)
{
    res2 = s2;
}
else
{
    res = -2;
    return "-2";
}
string s3=str.Substring(5,2);
if (s3 == "DN")
{
}
else
{
    res = -3;
```

```
    return "-3";
}
string s4 = str.Substring(7, 3);
Regex r3=new Regex(@"^([0-9]{3})$");
bool rs3 = r3.IsMatch(s4);
if (rs3)
{
   res3 = s4;
}
else
{
    res = -2;
    return "-2";
}
if (res == 0)
{
    sb.Append("Dotnet Batch ");
    sb.Append(res3);
    sb.Append(" has joined in ");
    sb.Append("20");
    sb.Append(res2);
    sb.Append(" year and is at ");
    sb.Append(res1);
```

```
            sb.Append(" Location");
        }
        string final = sb.ToString();
        return final;


    }
  }
}


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

namespace dummy
{
  class Program
  {
    static void Main(string[] args)
    {
      string str = Console.ReadLine();
      Console.WriteLine(Class1.checkcode(str));
```

```
        }
    }
}
```

ANOTHER METHOD:

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

namespace ConsoleApplication2
{
    class Usermaincode
    {
        public static string checkBatch(string input1)
    {
        Regex reg = new Regex(@"^(([A-Z]{3})+([0-9]{2})+[D][N]+([0-9]{3}))$");
            if (reg.IsMatch(input1))
            {
                string loc=input1.Substring(0,3);
                string bc = input1.Substring(7,3);
                string y = input1.Substring(3, 2);
```

```
        if (loc == "CHN" || loc == "CBE" || loc == "KOL" || loc == "PUN" || loc == "BGL" ||
loc == "KOC" || loc == "HYD")

        {

           if (loc == "CHN")

           {

              return "1";

           }

           else if (loc == "CBE")

           {

              return "2";

           }

           else if (loc == "KOL")

           {

              return "3";

           }

           else if (loc == "PUN")

           {

              return "4";

           }



           else if (loc == "BGL")

           {

              return "5";

           }
```

```
        else if (loc == "KOC")
        {
            return "6";
        }



        else if (loc == "HYD")
        {
            return "7";
        }
        else
        {
        }
    }
    else
    {  return "-1";
    }



}
else
{
```

```
            if(input1.Substring(5,2)!="DN")
            {
                return "-3";
            }
            else
                return "-2";
        }
        return "null";




        }
    }

}



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

```
        string input1 = Console.ReadLine();

        string bc = input1.Substring(7, 3);

        string y = input1.Substring(3, 2);

        string f = Usermaincode.checkBatch(input1);

        if (f == "1")

        {

            Console.WriteLine("DotNet batch " + bc + " has joined in 20" + y + " year and is at
Chennai Location");

        }

        else if (f == "2")

        {

            Console.WriteLine("DotNet batch " + bc + " has joined in 20" + y + " year and is at
Coimbatore Location");

        }



        else if (f == "3")

        {

            Console.WriteLine("DotNet batch " + bc + " has joined in 20" + y + " year and is at
Kolkata Location");

        }

        else if (f == "4")

        {

            Console.WriteLine("DotNet batch " + bc + " has joined in 20" + y + " year and is at
Pune Location");

        }
```

```
        else if (f == "5")
        {
            Console.WriteLine("DotNet batch " + bc + " has joined in 20" + y + " year and is at Bangalore Location");
        }
        else if (f == "6")
        {
            Console.WriteLine("DotNet batch " + bc + " has joined in 20" + y + " year and is at Kochi Location");
        }
        else if (f == "7")
        {
            Console.WriteLine("DotNet batch " + bc + " has joined in 20" + y + " year and is at Hyderabad Location");
        }
        else
        {
            Console.WriteLine(f);


        }


    }
  }
```

Duplicate Date Elements

Write a program to eliminate duplicate date elements in an input String Array. Print the resultant String array/list in dd/MM/yyyy format.

Business Rule:
1. Input Date will be of the form: dd-MM-yyyy or dd/MM/yyyy or dd.MM.yyyy or dd-Month-yyyy.
2. If any date is invalid, print 'Invalid Date'.

Create a class named UserProgramCode that has the following static method
public static List<string> **removeDuplicateDate(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 array.
The next 'n' lines of input consist of elements in the input array.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
5
20/09/2014
30-03-2015
13.06.2012
20-09-2014
20-September-2014

Sample Output 1 :
20/09/2014
30/03/2015
13/06/2012

Sample Input 2 :
5
20/09/2014
30-03-2015
13.13.2012
20-09-2014
20-September-2014

Sample Output 2 :
Invalid Date

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


namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            int num = int.Parse(Console.ReadLine());
            string[] r = new string[num];
            for (int i = 0; i < num; i++)
            {
                r[i] = Console.ReadLine();
            }
            List<string> res = UserProgramCode.removeDuplicateDate(r);
            foreach (string u in res)
            {
                Console.WriteLine(u);
            }
        }
    }
}
```

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

namespace ConsoleApplication2
{
    class UserProgramCode
    {
        public static List<string> removeDuplicateDate(string[] input1)
        {
            List<string> u = new List<string>();
            List<string> u1 = new List<string>();
            DateTime dt;
            string format = "dd-MM-yyyy";
            string format1 = "dd/MM/yyyy";
            string format2 = "dd.MM.yyyy";
            string format3 = "dd-MMMM-yyyy";
            foreach (string s in input1)
            {

                if (DateTime.TryParseExact(s, format, null,
System.Globalization.DateTimeStyles.None, out dt))
                {
                    u.Add(dt.ToString("dd/MM/yyyy"));
```

```
                    //Console.WriteLine(s);




                }


                else if (DateTime.TryParseExact(s, format1, null,
System.Globalization.DateTimeStyles.None, out dt))

                {


                    u.Add(dt.ToString("dd/MM/yyyy"));

                    //Console.WriteLine(s);

                }


                else if (DateTime.TryParseExact(s, format2, null,
System.Globalization.DateTimeStyles.None, out dt))

                {

                    u.Add(dt.ToString("dd/MM/yyyy"));

                   // Console.WriteLine(s);

                }


                else if (DateTime.TryParseExact(s, format3, null,
System.Globalization.DateTimeStyles.None, out dt))

                {

                    u.Add(dt.ToString("dd/MM/yyyy"));

                    //Console.WriteLine(s);


                }

                else
```

```
                {
                    u1.Add("Invalid Input");



                    return u1;
                }

            }
            u=u.Distinct().ToList();
            return u;



        }
    }
}
```

Identify Perfect Numbers

For a given integer input array ,write a program to identify the perfect numbers in the input array and store remaining elements excluding the perfect numbers in the output .Perfect number is a positive number in which the sum of all its positive divisors excluding that number is equivalent to that number itself.
Eg. 6 is a perfect number ,since its divisor are 1, 2 and 3.
Sum of its divisors is 1 + 2+ 3 = 6,which is equal to the number itself.

Business rule:
1) If any of the elements in input1 array is negative, then print -1.
2) If there are any duplicates found in input1 array, then print -2.
3) If size of the input1 array is 1 or greater than 7, then print -3.

Create a class named UserProgramCode that has the following static method
public static int[] perfectNum(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.
The next 'n' lines of input consist of elements in the input array.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
4
6
2
5
7

Sample Output 1 :
2
5
7

Sample Input 2 :
5
5
8
3
-4
6

Sample Output 2 :
-1

```
using System;

using System.Collections.Generic;

using System.Linq;
```

```
using System.Text;


namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int m = int.Parse(Console.ReadLine());
            int[] p = new int[m];
            for (int i = 0; i < m; i++)
            {
                p[i] =int .Parse( Console.ReadLine());
            }
            int[] r = UserProgramCode.findTriplets(p);
            foreach (int n in r)
            {

                Console.WriteLine(n);
            }



        }
    }
}
```

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


namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static int[] findTriplets(int[] b)
        {
            int sum;
            List<int> c=new List<int>();
            if(b.Length<=1 || b.Length>7)
            {
                c.Add(-3);
                return c.ToArray();
                }
            for (int t = 0; t < b.Length; t++)
            {
                for (int r = t + 1; r < b.Length; r++)
                {
                    if (b[t] == b[r])
                    {
                        c.Add(-2);
```

```
            return c.ToArray();
        }
    }
}



foreach (int a in b)
{
    if (a < 0)
    {
        c.Add(-1);
        return c.ToArray();
    }
    sum = 0;
    for (int i = 1; i < a; i++)
    {

        if (a % i == 0)
        {
            sum += i;
        }
    }
        if (sum == a)
        {
            c.Add(a);
        }
```

```
            }


            List<int> j = new List<int>(b);
            foreach (int u in c)
            {
                j.Remove(u);
            }
            return j.ToArray();
        }

    }
}
```

Remove Duplicates

Given a string as input, write a program to remove duplicate characters from the string.
Note - Only the first occurrence of each character should be retained. Retain all blank spaces.

Business Rule:
If there is no duplicate found, print -1.

Create a class named UserProgramCode that has the following static method
public static string removeDuplicates(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.
Refer business rules and sample output for formatting specifications.

Sample Input :
hi this is sample test

Sample Output :
hi ts ample

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;


namespace ConsoleApplication2

{

    class Program

    {

        static void Main(string[] args)

        {


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

```
            string res = UserProgramCode.removeDuplicateDate(r);


                Console.WriteLine(res);


        }
    }
}


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


namespace ConsoleApplication2
{
    class UserProgramCode
    {
        public static string removeDuplicateDate(string input1)
        {
            int p = 0;
            StringBuilder sb=new StringBuilder();
            char[] ch = input1.ToCharArray();
            for (int i = 0; i < ch.Length; i++)
            {
                for (int j = i + 1; j < ch.Length; j++)
                {
```

```
                    if (ch[i] == ch[j] && ch[i] != ' ')
                    {
                        ch[j] = '%';
                        p = 1;
                    }
                }
            }
            for (int i = 0; i < ch.Length; i++)
            {
                if (ch[i] != '%'&& p==1)
                {
                    sb.Append(ch[i]);
                }
                else
                {
                    return ("-1");
                }
            }
            return sb.ToString();

        }
    }
}
```

Count Vowels

Write a program to count the character which comes under the vowels sound from the given string .

The string value should have only the alphabet values.

Business Rules:

1. If the input string consists of any other character than the alphabets, return -1 from the method and print "Other characters found" in Main.

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

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

Input and Output Format:
Input is a string.
Output is an integer if the method returns the count, else a String "Other characters found" if the method returns '-1'.

Sample Input 1:
sang-gee-
Sample Output 1:
Other characters found

Sample Input 2:
god
Sample Output 2:
1

```
 using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;



namespace ConsoleApplication1

{

    class UserProgramCode
```

```
    {
        public static int countvowels(string input1)
        {    int count=0;
            char[] ch = input1.ToCharArray();
            foreach (char i in ch)
            {
                if (char.IsLetter(i))
                {
                    if (i == 'a' || i == 'e' || i == 'i' || i == 'o' || i == 'u' || i == 'A' || i == 'E' || i == 'I' || i == 'O' ||
i == 'U')
                    {
                        count++;
                    }
                }
                else
                    return -1;
            }
            return count;
        }
    }
}

class Program
    {
```

```
    static void Main(string[] args)
    {
        string input1 = Console.ReadLine();
        int c = UserProgramCode.countvowels(input1);
        if (c == -1)
            Console.WriteLine("Other Characters Found");
        else
            Console.WriteLine(c);
    }
}
```

Sort String

Given a string list as input{StringOne, StringTwo, StringThree,...}, Write a program to sort each character in the individual string in ascending order of alphabets {eginnOrSt, ginorStTw, eeghinrrStT,..}, then remove repetitive characters irrespective of case {eginOrSt, ginorStw, eghinrST,..}, then sort the resultant list in ascending order and print the array of string in lowercase{eghinrst,eginorst, ginorstw,..}. Ignore the case sensitivity of given input.

Business Rule:
1. If any of the elements in the given input contains any special characters or numbers, print 'Invalid Input'.
2. If there are duplicate elements in the given input,, remove the duplicates and follow all other steps.

Example1 :
input :
4
Ccaat
rat
dog
cow
output :

act
art
cow
dgo

Steps:
1. Sort each character of each string element: {aacCt,art,dgo,cow}
2. Remove repetitive characters irrespective of the case: {aCt,art,dgo,cow}
3. Now sort each string element and arrange it in ascending order in lowercase: {act,art,cow,dgo}

Example 2 :
input
4
cat
rat
dog1
cow$
output
Invalid Input

Example 3 :
input
5
cat
rat
cat
cow
rat
output
act
art
cow

Create a class named UserProgramCode that has the following static method
public static List<string> sortString(List<string> arr)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.
The next 'n' lines of input consist of elements in the input array.

Refer business rules and sample output for formatting specifications.

Sample Input :

4

Ccaat

rat

dog

cow

Sample Output :

act

art

cow

dgo

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

namespace sort_string
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = Convert.ToInt16(Console.ReadLine());
            string[] a = new string[n];

            for (int i = 0; i < n; i++)
            {
                a[i] = Console.ReadLine();
            }
            string[] res = Class1.string_sort(a);
            foreach (string j in res)
            {
                Console.WriteLine(j);
            }
        }

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

namespace sort_string
{
    class Class1
    {
        public static string[] string_sort(string[] s)
        {
            string[] str=new string[s.Length];
```

```
            string[] output = new string[1];
            int j = 0;
            foreach (string i in s)
            {
                char[] c = i.ToCharArray();
                foreach (char c1 in c)
                {
                    if (char.IsSymbol(c1)|(char.IsLetter(c1)))
                    {
                        output[0] = ("Invalid Input");
                        return output;
                    }
                }
                char[] ch = i.Distinct().ToArray();
                Array.Sort(ch);
                str[j] = new string(ch);
                j++;
            }
            Array.Sort(str);
            return str;
        }
    }
}
```

List the Elements - A

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

Example:
input1: [1,4,7,3,9,15,24]
input2: 17

Output1:[15,9,7,4,3,1]

Include a class **UserProgramCode** with static method **GetElements()** which accepts an integer list and the integer (input2) 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:
15
9
7
4
3
1

Sample Input 2:
6
5
9
3
4
16
21
9

Sample Output 2:
5
4
3

Largest Span

Write a program to read the size of the integer array and the elements of the array and find the size of largest Span in the given array. Print the output.

Note: Span is the number of elements between two repeated numbers including both numbers. Assume an array with single element has a span of 1.

Business rule:

If there is no number repeated in an array, return 0. If there are two repeated integers in the input array, consider the first number and return the span.

Example 1:

Input = {1, 2, 1, 1, 3}

Output = 4

Example 2:

Input = {1, 4, 2, 1, 4, 1, 5}

Output = 6

Include a class UserProgramCode with a static method getMaxSpan which accepts the size of the array and the integer array. The return type (integer) should return the span size.

Create a Class Program which would be used to accept the size of the array and the array elements and call the static method present in UserProgramCode.

Input and Output Format:
Input consists of n+1 integers, where the first integer corresponds to the size of the array followed by n integers .
Output consists of an integer(the span size).
Refer sample output for formatting specifications.

Sample Input 1:
5
1
2
1
1
3
Sample Output 1:
4

Sample Input 2:
7
1
4
2
1
4
1
5
Sample Output 2:
6

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;
```

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int m = int.Parse(Console.ReadLine());
            int[] p = new int[m];
            for (int i = 0; i < m; i++)
            {
                p[i] =int .Parse( Console.ReadLine());
            }
           // int n = int.Parse(Console.ReadLine());
            int r = UserProgramCode.findTriplets(p);
            if (r == -1)
            {
                Console.WriteLine("Invalid Input");
            }
            else
                Console.WriteLine(r);



        }
    }
}
```

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static int findTriplets(int[] b)
        {
            int c = 0,d=0;
            for (int i = 0; i < b.Length; i++)
            {
                for (int j = b.Length - 1; j > i; j--)
                {
                    if (b[i] == b[j])
                    {
                        c = j;
                        break;
                    }
                    else
                    {
                        d++;
                    }
                }
            }
            if (d >= b.Length)
            {
                return -1;
            }
            return c;


        }

    }
}
```

Find Leaders

Given an array of integer values as input,write a program that will fetch all the leaders in the array and print them after sorting them in ascending order.An element is a leader if it is greater than all

the elements to its right side. Consider that the rightmost element is always a leader.

Business Rules :
1. If the given input contains any negative number, then print -1.
2. If there are less than 2 elements or more than 10 elements in the input array,print -2.
3. If any of the elements in the input array are repetitive, then print -3.

Create a class named UserProgramCode that has the following static method
public static List<int> findLeadersArray(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.
The next 'n' lines of input consist of elements in the input array.
Refer sample output for formatting specifications.

Sample Input 1:
6
6
7
4
3
5
2

Sample Output 1:
2
5
7

Sample Input 2:
6
6
7
-4
3
5
2

Sample Output 2:
-1

```
class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            int[] a = new int[n];
            List<int> b = new List<int>();
            for (int i = 0; i < n; i++)
            {
                a[i] = int.Parse(Console.ReadLine());
            }
            b = leader.le(a);
            foreach (int x in b)
            {
                Console.WriteLine(x);
            }

        }
```

```
public static List<int> le(int[] a)
        {
            List<int> c = new List<int>();
            int l=a.Length;


            int count=0,d=0;
            for (int i = 0; i < l; i++)
            {
                for (int j = i + 1; j < l; j++)
                {
                    if (a[i] == a[j])
                    {
                        d++;



                    }
                }
                if (a[i] < 0)
                {
                    c.Add(-1);
                    return c;
                }
                if (l < 2 || l > 10)
```

```
    {
        c.Add(-2);
        return c;
    }

}
if (d != 0)
{
    c.Add(-3);

    return c;
}
    for (int i = 0; i < l-1; i++)
    {
        count = 0;
        for (int j = i + 1; j < l; j++)
        {

            if(a[i]<a[j])
            {
                count++;
            }
```

```
                }
                if(count==0)
                {
                    c.Add(a[i]);
                }
            }
        c.Add(a[l-1]);
            c.Sort();
        return c;
    }
```

ANOTHER:

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

namespace ConsoleApplication7
{
   class Usermaincode
   {
      public static List<int> findLeadersArray(int[] input1)
      {
         int c=0;
```

```
List<int> f=new List<int>();
List<int> li=new List<int>();
if (input1.Length < 2 || input1.Length > 10)
{
   f.Add(-2);
   return f;
}

foreach (int p in input1)
{
   if(p<0)
   {
      f.Add(-1);
      return f;
}
}
for (int i = 0; i < input1.Length-1; i++)
{   c=0;
   for (int j = i + 1; j < input1.Length; j++)
   {
      if (input1[i] == input1[j])
      {
         f.Add(-3);
```

```
            return f;
        }
        if (input1[i] > input1[j])
        {



        }
        else
        {
            c = 1;
        }
    }
    if (c==0)
    {
        li.Add(input1[i]);
    }
}
li.Add(input1[input1.Length-1]);
li.Sort();
return li;



}
```

```
    }
  }
```

```
class Program
  {
    static void Main(string[] args)
    {   int n = int.Parse(Console.ReadLine());
      int[] input1= new int[n];
      for (int i = 0; i < n; i++)
      {
        input1[i]=int.Parse(Console.ReadLine());
      }
      List<int> o = Usermaincode.findLeadersArray(input1);
      foreach (int u in o)
      {
        Console.WriteLine(u);
      }
    }
  }
```

Relative Order

Given two input integer arrays input1 and input2, write a program to sort input1 in such a way that the relative order among the elements will be same as those are in input2.
For the elements not present in input2,append them at last in sorted order.

Business Rules :
1. If any of the given inputs contains any negative number, then print -1.
2. If any of the elements in input 2 array is not available in input 1 array, then print -2.
3. If there are less than 3 elements or more than 15 elements in the input1 array, print -3.

Example
Input Array 1 = {2,1,2,5,7,1,9,3,6,8,8}
Input Array 2 = {2,1,8,3|
Output Array = {2,2,1,1,8,8,3,5,6,7,9}

Create a class named UserProgramCode that has the following static method
public static int[] relativeOrder(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, n that corresponds to the number of elements in the input array 1.
The next 'n' lines of input consist of elements in the input array 1.
The next line of the input consists of an integer, m that corresponds to the number of elements in the input array 2.
The next 'm' lines of input consist of elements in the input array 1.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
11
2
1
2
5
7
1
9
3
6
8
8
4
2

1
8
3

Sample Output 1 :
2
2
1
1
8
8
3
5
6
7
9

Sample Input 2 :
8
2
1
5
7
9
3
6
8
4
2
1
8
3

Sample Output 2 :
2
1
8
3
5
6
7
9

Sample Input 3 :

11
2
1
2
-5
7
1
9
3
6
8
8
4
2
1
8
3

Sample Output 3 :

-1

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


namespace ConsoleApplication4
{
    class Usermaincode
```

```
{
    public static int[] relativeOrder(int[] input1, int[] input2)
    {
        int temp = 0;
        List<int> li = new List<int>();

        List<int> m = new List<int>();
        int[] c = new int[1];
        if (input1.Length < 3 || input1.Length > 15)
        {
            c[0] = -3;
            return c;

        }
        for (int z = 0; z < input1.Length; z++)
        {
            if (input1[z] < 0)
            {
                c[0] = -1;
                return c;
            }
        }
        for (int x = 0; x < input2.Length; x++)
```

```
{
    if (input2[x] < 0)
    {
        c[0] = -1;
        return c;
    }
}
for(int d = 0; d < input2.Length ; d++)
{
    if (input1.Contains(input2[d]))
    {
    }
    else

        temp++;

}
if (temp >0)
{
    c[0] = -2;
    return c;
}
```

```
for (int i = 0; i < input2.Length; i++)
{
   for (int j = 0; j < input1.Length; j++)
    {

          if (input1[j] == input2[i])
          {
             li.Add(input1[j]);
             input1[j] = 0;
          }
    }
}



foreach (int u in input1)
{
    if (u > 0)
        m.Add(u);
}

m.Sort();
```

```
            li.AddRange(m);
            int[] o = li.ToArray();


            return o;


        }
    }
}



class Program
    {
        static void Main(string[] args)
        {
            int p = int.Parse(Console.ReadLine());
            int[] input1 = new int[p];
            for (int i = 0; i < p; i++)
            {
                input1[i] = int.Parse(Console.ReadLine());


            }
            int q = int.Parse(Console.ReadLine());
            int[] input2 = new int[q];
```

```
        for (int j = 0; j < q; j++)
        {
            input2[j] = int.Parse(Console.ReadLine());


        }
        int[] output = Usermaincode.relativeOrder(input1, input2);
        foreach (int t in output)
        {
            Console.WriteLine(t);
        }
    }
}
```

Get word with Maximum Vowels

Write a method that accepts a string input and returns the word with maximum number of vowels. In case there are two or more words with maximum number of vowels, return the first word.

Example:
Input: Appreciation is the best way to motivate
Output: Appreciation (total vowels = 6)

Include a class **UserProgramCode** with a static method **getWordWithMaximumVowels** that accepts a string and returns a string.

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

Input and Output Format:
Input consists of a string.
Output consists of a string.

Sample Input1 :
Appreciation is the best way to motivate

Sample Output1 :
Appreciation

Sample Input2 :
Sun rises in the east

Sample Output2 :
rises

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;



namespace ConsoleApplication1

{

    class Program

    {

        static void Main(string[] args)

        {

            string m = Console.ReadLine();

            string r = UserProgramCode.findTriplets(m);
```

```
                Console.WriteLine(r);


        }
    }
}
```

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static string findTriplets(string b)
        {
            int c,max=0;
            string n = "";
            string[] s=b.Split(' ');
            foreach (string m in s)
            {
                c = 0;
```

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

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

                {

                    if (ch[i] == 'a' || ch[i] == 'e' || ch[i] == 'i' || ch[i] == 'o' ||
ch[i] == 'u' || ch[i] == 'A' || ch[i] == 'E' || ch[i] == 'I' || ch[i] == 'O' || ch[i] ==
'U')

                    {

                        c++;

                    }







                }

                if (c > max)

                {

                    max = c;

                    n = m;



                }

            }



            return n ;



        }



    }

}
```

Calculate Take Home Salary

TMB Software solution is under developing software company in KolKatta. The company is providing the Provident fund for all the employees based on the salary criteria as given below. Write a method to find the take home salary for the employee. Medical Insurance for each employee is Rs 678.

Salary Range---------- PF

Lesser than 15000 ------- 750

15001 - 22000 -------- 850

22001-30000 -------- 925

Above 30000 -------- 1000

Take Home Salary = Salary - pf - MedicalInsurance

Include a class **UserProgramCode** with a static method **calculateHomeSalary** which accepts an integer and returns an integer that corresponds to the Take Home Salary.

The method returns -1 when the input integer is negative.

Create a class **Program** which would get the input and call the static
method **calculateHomeSalary** present in the **UserProgramCode**.
If the method returns -1, print 'Invalid Input'.

Input and Output format :
Input consists of a integer that represents a salary.
Output is an integer that corresponds to 'take home salary' or a string 'Invalid Input'.

Sample Input 1 :
13500
Sample Output 1 :
12072

Sample Input 2 :
-10000
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)

        {

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

            int r = UserProgramCode.findTriplets(m);

            if (r == -1)

            {

                Console.WriteLine("Invalid Input");

            }

            else

                Console.WriteLine(r);
```

```
        }
    }
}

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static int findTriplets(int k)
        {
            int p;
            if (k < 0)
            {
                return -1;
            }
            if (k < 15000)
            {
                p = k - 750 - 678;
            }
            else if (k > 15000 && k >= 22000)
```

```
            {
                p = k - 850 - 678;
            }
            else if (k > 22000 && k <= 30000)
            {
                p = k - 925 - 678;
            }
            else
                p = k - 1000 - 678;
            return p;



        }


    }
}
```

Unique Even Sum

Write a program to remove all duplicate elements from an input array and return the sum of all even numbers.

Example :
Input: {1,2,7,2,4,8,9,6,8}
After removing duplicates : {1,7,4,9,6}
Output: 4+6 = 10

Exception Rules:

If there is no even number in the input, return -1.

if input contains negative numbers, then return -2.

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

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 values. The first value corresponds to size of the array. The next n numbers contains the integer values.

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

Refer sample output for formatting specifications.

Sample Input 1:

9

1

2

7

2

4

8

9

6

8

Sample Output 1:

10

Sample Input 2:

5

1

3

5

7

9

Sample Output 2:

-1

Sample Input 3:

4

1

-2

6

8

Sample Output 3:

-2

```
using System;

using System.Collections.Generic;
```

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


namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int m = int.Parse(Console.ReadLine());
            int[] p = new int[m];
            for (int i = 0; i < m; i++)
            {
                p[i] =int .Parse( Console.ReadLine());
            }
           // int n = int.Parse(Console.ReadLine());
            int r = UserProgramCode.findTriplets(p);
            if (r == -1)
            {
                Console.WriteLine("Invalid Input");
            }
            else
                Console.WriteLine(r);



        }
```

```
    }
}


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


namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static int findTriplets(int[] b)
        {
            int sum=0;
            List<int>k=new List<int>();
            for (int i = 0; i < b.Length; i++)
            {
                for (int j = i + 1; j < b.Length; j++)
                {
                    if (b[i] == b[j])
                    {
                        b[i] = -2;
                        b[j] = -1;
                    }
```

```
            }
            if (b[i] > 0)
            {
                k.Add(b[i]);
            }

            if (b[i] < 0)
            {
                return -2;
            }
        }

        foreach (int e in k)
        {
            if (e % 2 == 0)
            {
                sum += e;
            }
            else
            {
                return -1;
            }
        }
        return sum;

    }
```

```
    }
}
```

Find nth Largest Number

Write a method to find the nth largest number in an input integer array.

Include a class **UserProgramCode** with a static method **findNthLargestNumber** which accepts 2 inputs, an integer array and an integer (n) and returns an integer.
If the input consists of any negative numbers, the method returns -1. Else the method returns the nth largest element in the array.

Create a class **Program** which would get the input and call the static method **findNthLargestNumber** present in the **UserProgramCode**.
If the method returns -1, then print 'Invalid Input'.

Input and Output Format:

The first line of the input consists of an integer that corresponds to m, the size of the array.
The next m lines of input consists of integers that correspond to the elements in the array.
The next line of input consists of integer that corresponds to 'n'.

Refer sample output for formatting specifications.

Sample Input 1:

7
2
1
67
10
55
12
7
-2

Sample Output 1:

Invalid Input

Sample Input 2:

7
100
300
150
450
650
50
25
4

Sample Output 2:

150

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;


namespace ConsoleApplication1

{
```

```
    class Program
    {
        static void Main(string[] args)
        {
            int m = int.Parse(Console.ReadLine());
            int[] p = new int[m];
            for (int i = 0; i < m; i++)
            {
                p[i] =int .Parse( Console.ReadLine());
            }
            int n = int.Parse(Console.ReadLine());
            int r = UserProgramCode.findTriplets(p,n);
            if (r == -1)
            {
                Console.WriteLine("Invalid Input");
            }
            else
                Console.WriteLine(r);



        }
    }
}

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

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


namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static int findTriplets(int[] b,int n)
        {
            if (n < 0)
            {
                return -1;
            }
            Array.Sort(b);
            Array.Reverse(b);
            int c = b[n-1];
            return c;
        }

    }
}
```

printCapitalized

Write a code to convert the first letter of each word to capital Case and return the final string

Example :
Input: ""Now is the time to act!""
Output: ""Now Is The Time To Act!""

Include a Class **UserProgramCode** with a static method **printCapitalized** which accepts a string as an input.The return type is String which is a sentence with first letter of each word capitalized.

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

Input and Output Format:
Input consists of string.
Output consists of a string which corressponds to first letter of each word to be capitalized and make other letters to be lower case

Sample Input 1:
Features Of JAVA2

Sample Output 1:
Features Of Java2

Sample Input 2:
gOogLe is A SeaRCh enGinEe

Sample Output 2:
Google Is A Search Enginee

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;



namespace ConsoleApplication1

{

    class Program

    {

        static void Main(string[] args)

        {
```

```
            string m = Console.ReadLine();
            string r = UserProgramCode.findTriplets(m);

                Console.WriteLine(r);

        }
    }
}

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static string findTriplets(string b)
        {
            string n="";
            string p = "";
            StringBuilder sb=new StringBuilder();
            string[] s = b.Split(' ');
            foreach (string m in s)
```

```
            {
                n = m.Substring(0, 1).ToUpper();
                p=m.Substring(1,m.Length-1).ToLower();
                sb.Append(n + p);
                sb.Append(" ");
            }
            return sb.ToString();




        }

    }
}
```

Add and Reverse

Given an int array and a number 'k' as input, write a program to add all the elements in the array greater than the given number 'k'. Finally reverse the digits of the obtained sum and print it.

Include a class **UserProgramCode** with a static method "**addAndReverse**" that accepts 2 arguments and returns an integer.The first argument corresponds to the integer array and the second argument corresponds to the number k.

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

Example:

Input Array = {10,15,20,25,30,100}

Number = 15

sum = 20 + 25 + 30 + 100 = 175

output = 571

Input and Output Format:

The first line of the input consists of an integer that corresponds to the number of elements in the array.

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

The last line of the input consists of an integer that corresponds to the number, k.

Output consists of a single integer.

Sample Input

6

10

15

20

25

30

100

15

Sample Output

571

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int m = int.Parse(Console.ReadLine());
            int[] p = new int[m];
            for (int i = 0; i < m; i++)
            {
                p[i] =int .Parse( Console.ReadLine());
            }
            int n = int.Parse(Console.ReadLine());
            int r = UserProgramCode.findTriplets(p,n);
```

```
            if (r == -1)
            {
                Console.WriteLine("Invalid Input");
            }
            else
                Console.WriteLine(r);


        }
    }
}

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static int findTriplets(int[] b,int n)
        {
            List<int> l=new List<int>();
            int sum=0,d,rev=0;
```

```
            for (int i = 0; i < b.Length; i++)
            {
                if (b[i] > n)
                {
                    sum += b[i];
                }
            }
            while (sum > 0)
            {
                d = sum % 10;
                rev = rev * 10 + d;
                sum = sum / 10;
            }

            return rev;

        }

    }
}
```

Concatenate Characters

Given an input string array, write a program to get the second character of each string and form a new String by concatenating the fetched characters together. Print the new string formed.

Business Rules :
1. If the given input array element contains numbers, then print -1.
2. If the given input array element contains special characters , then print -2.
3. If the input array contains only one string, then print -3.

Create a class named UserProgramCode that has the following static method
public static string concatCharacter(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 array .
The next 'n' lines of input correspond to elements in the input array.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
3
ab
aaa
adbcd

Sample Output 1 :
bad

Sample Input 2 :
4
ban
b%a
ssm
tea

Sample Output 2 :
-2

Count Subsets

Given a method with an integer list as input, Write code to find the number of subsets formed from the given input. Consider any three elements for the input list which forms as one subset. Sum of first two elements must be equal to third element. The number of subsets which satisfy these conditions would be the output that is printed

Note: The elements in a subset should satisfy below conditions:
1)Any subset should have only 3 elements
2)The elements in each subset must be distinct

Business rule:
1) Print -1 when no such subsets are formed
2) Print -2 if input list consists of negative elements
3) Print -3 when same integer element is repeated twice in the input list.

Example:
input
5
Input1
1
2
3
4
6
The subsets formed are (1,2,3), (1,3,4), (2,4,6)
output =
3

Create a class named UserProgramCode that has the following static method
public static int countSubsets(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.
The next 'n' lines of input consist of elements in the input array.
Output consists of an integer.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :

5

1

2

3

4

6

Sample Output 2 :

3

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

namespace ConsoleApplication8
{
    class Program
    {
        static void Main(string[] args)
        {
            int n =int.Parse( Console.ReadLine());
            int[] b = new int[n];
            for (int i = 0; i < n; i++)
            {
                b[i] = int.Parse(Console.ReadLine());
            }

            Console.WriteLine(Class1.countsubsets(b));
        }
    }
}


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

namespace ConsoleApplication8
{
    class Class1
    {
        public static int countsubsets(int[] a)
        {
            int c = 0;
            for (int i = 0; i < a.Length; i++)
            {
                for (int j = i + 1; j < a.Length; j++)
                {
                    if (a[i] < 0)
                    {
                        return -2;
```

```
                    }
                    else if (a[i] == a[j])
                    {
                        return -3;
                    }
                    for (int k = i+2; k < a.Length; k++)
                    {
                        if (a[i] + a[j] == a[k])
                            {
                                c++;
                            }
                            else
                            {
                            }
                    }
                }
            }

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



        }
    }
}
```

Roman Numeral

Given an integer as input, write a program to convert integer input to roman numerals . Represent it as a string.

Basic Steps for Roman number calculation:

1. I is the numeral one.
V is the numeral 5.
X is the numeral 10.
L is the numeral 50.
C is the numeral 100.
D is the numeral 500.
M is the numeral 1000.

2.A smaller number in front of a larger number means subtraction, everything else means addition. For example, IV means 4, VI means 6.
You would not put more than one smaller number in front of a larger number to subtract. For example, IIV would not mean 3.
You must separate ones, tens, hundreds, and thousands as separate items. This means that 99 is XCIX, 90 + 9, but never should be written as IC. Similarly, 999 cannot be IM and 1999 cannot be MIM.
So, II is two, III is three. VII is 7, VIII is 8. IX is 9, XI is 11, etc. Again, XL would be 40, LX would be 60, LXX would be 70, LXXX would be 80 etc. Similarly, XC would be 90, XCIX would be 99, CL would be 150, CLIX would be 159, CXC would be 190, CC would be 200, CCC would be 300, etc. Again, CD would be 400, DC would be 600, etc. And CM would be 900.

Business Rule:
1. If the input is less than 0, then print "Invalid Input" .
2. If the given input variable is greater than 4000, then print "Greater Than 4000" .

Create a class named UserProgramCode that has the following static method
public static string romanNumerals(int input1).
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
Input consists of an integer.
Output is a string.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
2086

Sample Output 1 :
MMLXXXVI

Sample Input 2 :
2091

Sample Output 2 :
MMXCI

Sample Input 3 :
-2091

Sample Output 3 :

Invalid Input

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

namespace ConsoleApplication4
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            string res = UserProgramCode.sample1(n);
            Console.WriteLine(res);
        }
    }
}

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

namespace ConsoleApplication4
{
    class UserProgramCode
    {
        public static string sample1(int n)
        {
            StringBuilder sb = new StringBuilder();
            string s1 = "";
            int[] num = { 1, 4, 5, 9, 10, 40, 50, 90, 100, 400, 500, 900, 1000 };
            string[] s = { "I", "IV", "V", "IX", "X", "XL", "L", "XC", "C", "CD", "D",
"CM", "M" };
            if (n < 0)
            {
                s1 = "Invalid Input";
            }
            else if (n > 4000)
            {
                s1 = "Greater than 4000";
            }
            else
            {
                while (n > 0)
                {
                    for (int i = num.Length - 1; i >= 0; i--)
                    {
                        if (n / num[i] >= 1)
                        {
                            n = n - num[i];
                            sb.Append(s[i]);
                            break;
                        }
                    }
                }
```

```
            }
            s1 = sb.ToString();
            return s1;
        }
    }
}
```

Validate Voter

Write a program to Validate the eligibility of the users for Voting in Election by accepting the input as Date Of Birth and the Date Of Election. Accept the dates as strings. Validate the DOB against the Date of Election, if it is atleast 18 yrs.

Business Rules:

1.Only date format “mm/dd/yyyy” should be given as input, if not return -1.

2. The eligible voting age is from 18 years.(Including 18)

3. If the age is valid for voting , then return 1.

4. If the age is invalid for voting , then return 0.

Include a class UserProgramCode with a static method validateVoter which accepts two Strings. The return type (Integer) should return a value according to the business rules.

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

Input and Output Format:

Input consists of two Strings, the first String corresponds to the DOB and the second String corresponds to the Date Of Election.

Output consists of a String, (“Invalid Date format” if -1 is returned, “Eligible” if 1 is returned, “Not Eligible” if 0 is returned.

Refer sample output for formatting specifications.

Sample Input 1:

12/29/1990

09/11/2014

Sample Output 1:

Eligible

Sample Input 2:

12/29/2010

09/11/2014

Sample Output 2:

Not Eligible

Sample Input 3:

32/29/1990

09/11/2014

Sample Output 3:

Invalid Date format

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

namespace ConsoleApplication9
{
    class Program
    {
        static void Main(string[] args)
        {
            string a = Console.ReadLine();
            string b = Console.ReadLine();

            int res = Class1.validatevoter(a, b);
            if (res == 1)
            {
                Console.WriteLine("Eligible");
            }
            else if (res == 0)
            {
                Console.WriteLine("Not Eligible");
            }
            else if (res == -1)
            {
                Console.WriteLine("Invalid Date Format");
            }

        }
    }
}

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

namespace ConsoleApplication9
{
    class Class1
```

```
    {
        public static int validatevoter(string a, string b)
        {
            DateTime dt;
            DateTime dt1;
            bool
c=DateTime.TryParseExact(a,"MM/dd/yyyy",null,System.Globalization.DateTimeStyles.None,out
dt);
            bool
d=DateTime.TryParseExact(b,"MM/dd/yyyy",null,System.Globalization.DateTimeStyles.None,out
dt1);

            if (c && d)
            {
                if (dt1.Year - dt.Year > 17)
                {
                    return 1;
                }
                else
                {
                    return 0;

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

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

Validate String

For a given String apply the following validations.

1. The given input String should be only four characters long. If not print -1.
2. First character can be an alphabet or digit. If not print -2 .
3. Second character must be uppercase alphabet. (eg 'M','R'.. any alpbabet A - Z). If not print -3 .
4. Third character must be a number and also between 5-9. If not print -4 .
If all the conditions are satisfied print 1.

Example 1:
input='vM7u3'
output = -1

Example 2:
input='&Mau'
output = -2

Example 3:
input='vrau'
output = -3

Example 4:
input='vR3a'
output = -4

Example 5:
input='vR5a'
output = 1

Create a class named UserProgramCode that has the following static method
public static int validateString(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 :
vR5a

Sample Output :
1

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


namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            string q = Console.ReadLine();


            int op = UserProgramCode.replaceChar(q);
            Console.WriteLine(op);
        }
    }
}


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

```
using System.Text.RegularExpressions;


namespace ConsoleApplication3
{
    class UserProgramCode
    {
        public static int replaceChar(String s)
        {
            if (s.Length > 4)
            {
                return -1;
            }
            Regex reg = new Regex(@"^(([A-Za-z0-9]{1})+([A-Z]{1})+([5-9]{1})+([A-Za-z0-
9]{1}))$");
            if (reg.IsMatch(s))
            {
                return 1;
            }
            char[] ch = s.ToCharArray();
            if (!(ch[0] >= 'a' && ch[0] >= 'z' || ch[0] >= 'A' && ch[0] >= 'Z' || ch[0]
>= 0 && ch[0] <= 9))
            {
                return -2;
            }
            else if (!(ch[1] >= 'A' && ch[1] <= 'Z'))
            {
                return -3;
```

```
            }
            else if (!(ch[2] >= 5 && ch[2] <= 9))
            {
                return -4;
            }









            else
                return 0;


        }
    }
}
```

Vowels

A string is said to be valid if it contains exactly five vowels in any order. Assume there is no repetition of any vowel in the given string.

Example:
Input : acbisouzze
Output: Valid

Include a class **UserProgramCode** with a static method **testVowels that accepts a string and returns an integer. The method returns 1 if the string is valid. Else it returns -1.**

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

If there are exactly five vowels present in the string then print "Valid" else print as "Invalid".

Sample Input 1:
education

Sample Output 1:
Valid

Sample Input 2:
vowels

Sample Output 2:
Invalid

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;



namespace vowels

{

    class Program
```

```
    {
        static void Main(string[] args)
        {
            string str = Console.ReadLine();
            int res=(Class1.vowels(str));
            if(res==1)
                Console.WriteLine("Valid");
            else
                Console.WriteLine("Invalid");
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace vowels
{
    class Class1
    {
        public static int vowels(string str)
        {
```

```
        StringBuilder sb = new StringBuilder();
        char[] ch = str.ToCharArray();
        foreach (char i in ch)
        {
            if (i == 'a' || i == 'e' || i == 'i' || i == 'o' || i == 'u')
                sb.Append(i);


        }
        char[] a = sb.ToString().ToCharArray();
        char[] b=a.Distinct().ToArray();
        if (b.Length != 5)
            return -1;
        else
            return 1;
    }
  }
}
```

Display Students Exam Eligibility status

A university has the following rules for a student to qualify for a degree with A as the main subject and B as the subsidiary subject:

Business Rule:

(a) He/She should get 55 percent or more(>=55%) in A and 45 percent or more(>=45%) in B.

(b) If he/she gets less than 55(<55%) percent in A he/she should get 55 percent or more(>=55%) in B. However, he/she should get at least 45 percent(>=45%) in A.

(c) If he/she gets less than 45 percent(<45%) in B and 65 percent or more(>=65%) in A he/she is allowed to reappear in an examination in B to qualify.

(d) In all other cases he/she is declared to have failed.

Write a code to display the student status according to above rules. Consider inputs as marks for both the subject.

Include a class **UserProgramCode** with static method **FindResult()** that accepts two integers. The return type should be String.

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

Input and Output Format :

Input1- % of Marks in A

Input2- % of Marks in B

Output1- Result("P" for Pass ,"F" for fail ; "R" for Reappear)

If the input is more than 100 return "Invalid Input" from **FindResult**(**int input1,int input2**) else return appropriate result.

Then display the result in **Program** class.

Sample Input 1:
42
45
Sample Output 1:
F

Sample Input 2:
105
05
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)

        {

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



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

            string r = UserProgramCode.findTriplets(m,n);
```

```
            Console.WriteLine(r);



        }
    }
}


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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static string findTriplets(int a,int b)
        {
            if (a > 100 || b > 100)
            {
                return "Invalid Input";
            }
            if (a >= 55 && b >= 45)
            {
                return "P";
```

```
            }
            else if (a >= 45 && b >= 55)
            {
                return "P";
            }

            else if (b < 45 && a >= 65)
            {
                return "R";
            }
            else
                return "F";

        }

    }
}
```

Calculate VAT

ABC stores needs a computerized solution for calculating the VAT for the billing amount. Write a code to calculate the VAT amount for the input bill amount. The VAT should be calculated in the following basis,

Type ------- VATPercentage,

Medical (M) - - - - - - - - - -- 9%

Vegetables (V) and fruits - - - - - - - - - - - 5%

Clothes (C) - - - - - - - - - - - - 12%

Electronics (E) - - - - - - - - - - -- - - 6.25%

Business Rules :

1. The codes 'M','V','C' or 'E' only should be given as input for indicating the Medical, Vegetables and Fruits, Clothes and Electronics type of goods respectively. Any other character other than the above is given as input, it is Invalid Input.

2. Only Positive number should be given as a input for bill amount. Else it is Invalid Input.

Include a class **UserProgramCode** with a static method **calculateVAT** which accepts a character and double and returns a double which corresponds to the calculated VAT amount. If the input is invalid, the method returns -1.

Create a class **Program** which would get the input and call the static method **calculateVAT** present in the **UserProgramCode**.
If the method returns -1, print 'Invalid Input'.

Input and Output Format:

Input consists of character and double.
Character denotes a goods type and double denotes total amount.

Refer sample output for formatting specifications.

Sample Input 1 :

M
70
Sample Output 1 :
6.3

Sample Input 2 :
V
-500
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)

        {

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


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

            double r = UserProgramCode.findTriplets(m,n);

            Console.WriteLine(r);
```

```
        }
    }
}

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static double findTriplets(char a,double b)
        {
            double p=0;
            if (b < 0)
            {
                return -1;
            }
            if (a == 'M')
            {
                p = b * 0.09;

            }
```

```
            else if (a == 'V')
            {
                p = b * 0.05;


            }
            else if (a == 'C')
            {
                p = b * 0.12;
            }
            else if (a == 'E')
            {
                p = b * 0.0625;
            }
            else
                return -1;
            return p;
        }


    }
}
```

Quadratic Equation

Consider two equations x^2 - 2y + z = 0 and x^2+y=40 . Given an input integer array input1 containing values for the variable 'x', write a program to apply the input values to the equations and find out the corresponding y and z values to store them in the output in the following format (y1,z1,y2,z2,....} and so on. Print the output array
Output array elements can contain negative values.

Business rule:
1) If any of the elements in input1 array is negative, then print -1.
2) If there are any duplicates found in input1 array, then print -2.
3) If size of the input1 array is 1 or greater than 10, then print -3.

Example 1:
input1 : {1,2,4,5}
output1: {39,77,36,68,24,32,15,5}

Example 2:
input1 : {5,8,3,-4,6}
output1: {-1}

Create a class named UserProgramCode that has the following static method
public static int[] quadEquation(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.
The next 'n' lines of input consist of elements in the input array.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
4
1
2

4
5

Sample Output 1 :
39
77
36
68
24
32
15
5

Sample Input 2 :
5
5
8
3
-4
6

Sample Output 2 :
-1

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;


namespace Quadratic_number

{

    class Program

    {
```

```
        static void Main(string[] args)
        {
            int n = Convert.ToInt16(Console.ReadLine());
            int[] a = new int[n];
            for (int i = 0; i < n; i++)
               {
                  a[i] = Convert.ToInt16(Console.ReadLine());
               }

            int[] b = UserProgramCode.quadEquation(a);
            foreach(int c in b)
            {
               Console.WriteLine(c);
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace Quadratic_number
{
    class UserProgramCode
    {
        public static int[] quadEquation(int[] input1)
        {
            int[] output = new int[2*input1.Length];
            int[] out1 = new int[1];
            int n = input1.Length;
            if (n == 0 || n > 10)
            {
                out1[0] = -3;
                return out1;
            }
            int j=0;
            for (int i = 0; i < input1.Length; i++)
            {
                if (input1[i] < 0)
                {
                    out1[0] = -1;
                    return out1;
                }
```

```
        }


        int[] a = input1.Distinct().ToArray();
        if (input1.Length > a.Length)
        {
          out1[0]=-2;
          return out1;
        }
        for (int k = 0; k < input1.Length; k++)
        {
            output[j] = 40 - (input1[k] * input1[k]);
            j++;
            output[j ] = 80 -3*(input1[k] * input1[k]);
            j++;
        }
        return output;
      }
   }
}
```

Reimbursement

Hina University offers fees reimbursement scheme to its students according to the percentage they secure in board examinations, as per the below criteria.

Category A: For 80% to 85% (inclusive of border values) secured in exam: Refundable amount is 40% of the fees paid by the student during the start of the academic year and a cash award.

Category B: For 86% to 90% (inclusive of border values) secured in exam: Refundable amount is 50% of the fees paid by the student during the start of the academic year and a cash award.

Category C: For above 90% : Refundable amount is 60% of the fees paid by the student during the start of the academic year and a cash award.

The University also awards the students with a cash prize of Rs. 3000, Rs. 5000 and Rs. 7000 for Categories A, B and C respectively.
However, for the student to be eligible for reimbursement, he should NOT have a backlog (arrear) in any subject.

Write a program that calculates the total amount the student receives from the University which includes refundable fee amount and cash prize, according to:

1. The fees he pays at the start of the academic year, (this is the first input).
2. His/Her percentage of marks in the exams and (this is the second input).
3. His backlog status(Boolean : True if there is arrear and False if there is no arrear, this is the third input).
(Total amount the student receives = Refundable fee amount + Cash Prize)

Validation Rules :

1. Only positive number greater than 25000 should be given for fees amount else return -1.
2. Only numbers in the range 80 to 100 should be for percentage else return -2.

Include a class UserProgramCode with a static method calulateAmountRefundable which accept the fees, mark percentage and the backlog status. The return type (integer) should return the Refundable amount, or -1, or -2, accordingly.

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 double ,integer and a boolean value, where double corresponds to the fees, integer corresponds to the percentage and the boolean values corresponds to the backlog status.
Output consists of an Integer or one of the 2 strings ("Low fees amount" or "Invalid percentage").

Refer sample output for formatting specifications.

Sample Input 1:
25000
82
false
Sample Output 1:
13000

Sample Input 2:
20000
82
false
Sample Output 2:
Low fees amount

Sample Input 3:
20000
72
false
Sample Output 3:
Invalid percentage

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;



namespace ConsoleApplication1

{

    class Program

    {

        static void Main(string[] args)

        {

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



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

            bool val = bool.Parse(Console.ReadLine());

            int r = UserProgramCode.findTriplets(m,n,val);
```

```
            Console.WriteLine(r);



        }
    }
}


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


namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static int findTriplets(double a,int b,bool c)
        { double z=0;


        if (b < 80 || b > 100)
        {
            return -2;
        }
        if (a < 25000)
        {
```

```
            return -1;
        }
        if (b >= 80 && b <= 85 &&c==false)
        {
            z = (a * 0.4) + 3000;
        }
        else if (b >= 86 && b <= 90 && c == false)
        {
            z = (a * 0.5) + 5000;
        }
        else if (b > 90 && c == false)
        {
            z = (a * 0.6) + 7000;
        }
        return Convert.ToInt16(z);


        }


    }
}
```

Image Types

Given a string array input which consists of image file names along with their respective image type extensions in the format ("filename.extensiontype",..so on). The image file name and the extension are seperated by a dot (.)operator.
Write a program to calculate the count of image files having same extension type and store the values in the output string array variable in the below format.
output (Key,Value) = (ExtensionType1,count1,ExtensionType2,count2,...so on) .
Output should be stored in descending order based on the count of image files having the same extension type.

Note: jpeg,jfif,exif,tiff,raw,gif,bmp,png are the various types of image file extensions

Business Rules:
1)If all the elements of the input array do not have image type extension, then print -1.
2)If any of the file name doesn't contain extension type or if the extension is not an image type then it will be treated as other type, take the count of all such files and store as the last element in the sorted output array with key element as "others" and value element as the calculated count.
3)If more than one key element have same count, then store the key and their respective value element in the order given in input.

Create a class named UserProgramCode that has the following static method
public static List<string> countImageTypes(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 array .
The next 'n' lines of input correspond to elements in the input array.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
4
Employee.jpeg
Purchase.jpeg
stock.jpeg
book.gif

Sample Output 1 :
jpeg
3
gif

1

Sample Input 2 :
7
Sales.doc
Employee.jpeg
Purchase.jpeg
image.png
stock.jpeg
book.gif
pen

Sample Output 2 :
jpeg
3
png
1
gif
1
others
2

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;



namespace ConsoleApplication6

{

    class Usermaincode

    {

        public static List<string> countImageTypes(string[] input1)
```

```
    {
        int[] c = new int[9];
        List<string> v = new List<string>();
        List<string> i = new List<string>();
        List<string> o = new List<string>();
        foreach (string s in input1)
        {
            i = s.Split('.').ToList();
            foreach (string m in i)
            {
                v.Add(m);
            }
        }
        int k = 0;
        for (int l = 0; l < v.Count; l++)
        {
            if ((l % 2) != 0)
            {
                if (v[l] == "jpeg" || v[l] == "jfif" || v[l] == "exif" || v[l] == "raw" || v[l] == "tiff" || v[l]
== "gif" || v[l] == "bmp" || v[l] == "png")
                {
                    if (v[l] == "jpeg")
                    {
```

```
    k = 0;
    c[k]++;
}

else if (v[l] == "jfif")
{
    k = 1;
    c[k]++;
}
else if (v[l] == "exif")
{
    k = 2;
    c[k]++;
}
else if (v[l] == "raw")
{
    k = 3;
    c[k]++;
}
else if (v[l] == "tiff")
{
    k = 4;
```

```
        c[k]++;
    }
    else if (v[l] == "gif")
    {
        k = 5;
        c[k]++;
    }
    else if (v[l] == "bmp")
    {
        k = 6;
        c[k]++;
    }
    else if (v[l] == "png")
    {
        k = 7;
        c[k]++;
    }
}

else
{
    k = 8;
```

```
            c[k]++;
        }
    }
}

for(int u=0; u<v.Count; u++)
{
    if ((u % 2) != 0)
    {
        if (v[u] == "jpeg")
        {
            if (!o.Contains(v[u]))
            {
                o.Add(v[u]);
                o.Add(c[0].ToString());
            }
        }

        else if (v[u] == "jfif")
        {
            if (!o.Contains(v[u]))
            {
```

```
        o.Add(v[u]);
        o.Add(c[1].ToString());
    }
}

else if (v[u] == "exif")
{
    if (!o.Contains(v[u]))
    {
        o.Add(v[u]);
        o.Add(c[2].ToString());
    }
}
else if (v[u] == "raw")
{
    if (!o.Contains(v[u]))
    {
        o.Add(v[u]);
        o.Add(c[3].ToString());
    }
}
else if (v[u] == "tiff")
```

```
{
    if (!o.Contains(v[u]))
    {
        o.Add(v[u]);
        o.Add(c[4].ToString());
    }
}
else if (v[u] == "gif")
{
    if (!o.Contains(v[u]))
    {
        o.Add(v[u]);
        o.Add(c[5].ToString());
    }
}
else if (v[u] == "bmp")
{
    if (!o.Contains(v[u]))
    {
        o.Add(v[u]);
        o.Add(c[6].ToString());
    }
```

```
        }
        else if (v[u] == "png")
        {
            if (!o.Contains(v[u]))
            {
                o.Add(v[u]);
                o.Add(c[7].ToString());
            }
        }
        else
        {
            if (!o.Contains("Other"))
            {
                o.Add("Other");
                o.Add(c[8].ToString());
            }
        }
    }
}

return o;
```

```
        }
    }
}


 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();
            }
            List<string> li = Usermaincode.countImageTypes(str);
            foreach (string h in li)
            {
                Console.WriteLine(h);
```

```
        }

    }

}
```

Reverse and Format

Write a program to read a String and a character and to reverse the string and return it in a format such that each character is separated by the given character. Print the final string.

Example:
input1: "Rabbit"
input2: '-'
output: "t-i-b-b-a-R"

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

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

Input and Output Format:
Input consists of a string and a character.
Output consists of a String(the final output).
Refer sample output for formatting specifications.

Sample Input:
Rabbit
-
Sample Output:
t-i-b-b-a-R

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;
```

```
namespace Reverse_and_format
{
    class Program
    {
        static void Main(string[] args)
        {
            string str = Console.ReadLine();
            char c = Convert.ToChar(Console.ReadLine());
            Console.WriteLine(Class1.RABBIT(str,c));

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

namespace Reverse_and_format
{
    class Class1
```

```
    {
        public static string RABBIT(string str,char c)
        {
            StringBuilder sb = new StringBuilder();
            char[] ch = str.ToCharArray();
            Array.Reverse(ch);
           for(int i=0;i<ch.Length-1;i++)
            {
                sb.Append(ch[i]);
                sb.Append(c);
            }
           sb.Append(ch[ch.Length-1]);
           string s = sb.ToString();
           return s;


        }
    }
}
```

Fixed Point

Given an input array of n distinct integers sorted in ascending order, write a program that finds a Fixed Point in the array .
Fixed Point in an array is an index i such that arr[i] is equal to i.

Business Rules :
1. If any of the given inputs contain any negative number, then print -1.

2. If there are no fixed point values found in the input array, then print -2.
3. If there are less than 2 elements or more than 10 elements in the input array, then print -3.

Assume that there will be a maximum of 1 fixed point in the input array.

Create a class named UserProgramCode that has the following static method
public static int findFixedpoint(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
The first line of the input consists of an integer, n that corresponds to the number of elements in the input array 1.
The next 'n' lines of input consist of elements in the input array 1.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
6
1
4
45
3
0
19

Sample Output1 :
3

Sample Input 2 :
5
1
10
5
2
-7

Sample Output 2 :
-1

String Occurences

Write a program to count the number of occurences of second word of second sentence in the first sentence.

Return the count as output.
Note - Consider case.
Include a class **UserProgramCode** with a static method **countNoOfWords** which accepts two string variables. The return type is the integer.

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

Input and Output Format:

Input consists of two strings with maximum size of 100 characters.

Output consists of an integer.

Refer sample output for formatting specifications.

Sample Input 1:

abc bcd abc bcd abc abc

av abc

Sample Output 1:

4

Sample Input 2:

ABC xyz AAA

w abc

Sample Output 2:

0

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;
```

```
namespace string_occurances
{
    class Program
    {
        static void Main(string[] args)
        {
            string s1 = Console.ReadLine();
            string s2 = Console.ReadLine();
            Console.WriteLine(userprogram.countNoOfWords(s1, s2));

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

namespace string_occurances
{
```

```
class userprogram
{
    public static int countNoOfWords(string s1,string s2)
{
    int count = 0;
        string[] str = s2.Split();
        for(int i=0;i<s1.Length-1;i++)
        {
            for(int j=1;j<s1.Length-i-1;j++)
            {
                string s = s1.Substring(i,j);
                if(str[1]==s)
                {
                    count++;
                }
            }
        }
        return count;
}
}
}
```

Calculate Cost

Samira Florists take orders for flower decoration in social events and functions. They charge their customers based on:

The weight of flowers (kg) and
Decoration Options – Simple (S), Customized (C).
Flower type - Normal flowers (N) cost Rs.400 per kg
Exotic flowers (E) cost Rs. 700 per kg

They charge an additional Rs.15000 for simple decoration and Rs.25000 for a Customized decoration, apart from the cost of flowers. Also, they take only a minimum order of Rs. 20000 or more (including the flower cost and the decoration cost).

Business Rule:

If the cost calculated is less Rs 20000, then return -1.
If the type of flowers given is other than (N) or (E), then return -2.
If the type of decorations given is other than (S) or (C), then return -3.

Write a program to read an Integer and two characters, and calculate the total amount the customer pays to the florists. The values are: weight of flowers(in kg), the type of flowers (N or E)and the type of decoration(S or C). Print the final cost, or print “Too low cost”, if method returns -1, print “Invalid type of flower”, if method returns -2, print “Invalid decoration type”, if the method returns -3.

(Total Amount customer pays = Cost of flowers + Cost of Decoration)

Include a class UserProgramCode with a static method calculateCost which accept an integer and two characters . The return type (integer) should return output according to the business rules.

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

Input and Output Format:
Input consists of an integer, which corresponds to the weight, and two characters, which correspond to the flowers type and decoration type respectively.
Out put consists of an integer or a string.
Refer sample output for formatting specifications.

Sample Input 1:
10
N
B
Sample Output 1:
Invalid decoration type

Sample Input 2:
30
A
S
Sample Output 2:
Invalid type of flower

Sample Input 3:
10
N
S
Sample Output 3:
Too low cost

Sample Input 4:
20
N
S
Sample Output 4:
23000

Count the number of odd integers

Write a code to count the number of odd integers in the given integer array.

Include a class **UserProgramCode** with static method **countOddIntegers** that accepts an integer array and the return type should be int (count of odd Integers). Return -1 if the array contains negative values.
Create a class **Program** which would get the input and call the static method **countOddIntegers** present in the**UserProgramCode**.
In **Program** display “The Array consists non-positive value(s)” if -1 is returned, else print the count.

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 or a string.
Refer sample output for formatting specification.

SAMPLE INPUT 1:
2
-1
2
SAMPLE OUTPUT 1:
The Array consists non-positive value(s)

SAMPLE INPUT 2:
2
1
3
SAMPLE OUTPUT 2:
2

Dash Check

Write a program to read two strings and check whether or not they have dashes in the same places. Print "Yes" if the condition satisfies, else print "No".
Include a class **UserProgramCode** with a static method **compareDashes** which accepts two strings and returns an integer. The function returns 1 if all dashes are placed correctly, else the function returns 2.

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

Note: The strings must have exactly the same number of dashes in exactly the same positions. The strings might be of different length.
Input and Output Format:

Input consists of two strings.

Output consists of a string ("Yes" or "No").

Refer sample output for formatting specifications.

Sample Input 1:
hi—there-you.
12--(134)-7539
Sample Output 1:
Yes

Sample Input 2:
-15-389
-xyw-zzy
Sample Output 2:
No

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

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

namespace ConsoleApplication10
{
    class Program
    {
        static void Main(string[] args)
        {
            string a = Console.ReadLine();
            string b = Console.ReadLine();

            int res = Class1.dashcheck(a,b);
            if (res == 1)
            {
                Console.WriteLine("Yes");
            }
            else
            {
                Console.WriteLine("No");
            }
        }
    }
}
```

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

namespace ConsoleApplication10
{
    class Class1
    {
        public static int dashcheck(string a, string b)
        {
            int e = 0;
            string[] c = a.Split('-');
            string[] d = b.Split('-');

            if (c.Length == d.Length)
            {
                for (int i = 0; i < c.Length-1; i++)
                {
                    if (c[i].Length == d[i].Length)
                    {
                        e++;
                    }
                    else
                    {
                        return 2;
                    }
                }
            }
            else
            {
                return 2;
            }
```

```
            if (e == c.Length-1)
            {
                return 1;
            }
            else
            {
                return 2;
            }


        }
    }
}
```

Common Characters

Write a method to count the common character from the given two Strings.

Rule:
- Space should not be counted as a letter.
- Consider letters to be case sensitive. ie, 'a' is not equal to 'A'.

Example:
input1 = ""a black cow""
input2 =""battle ship""
output = 3

Include a class **UserProgramCode** with static method **commonChars** which accepts two String values.The return type is an interger.

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

Input Output format:
The input consists of two strings.
The output is an interger which counts the common characters.

Sample input 1:
a black cow
battle ship
Sample Output 1:
3

Sample input 2:
australia
sri lanka
Sample Output 2:
5

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;



namespace ConsoleApplication4

{

    class Program

    {

        static void Main(string[] args)

        {

            string str = Console.ReadLine();

            string str1 = Console.ReadLine();

            Console.WriteLine(Class1.commonChars(str,str1));







        }

    }

}
```

```
class Class1
  {
      public static int commonChars(string str, string str1)
      {
          int j = 0; int k = 0; int count = 0;
          char[] ch = str.ToCharArray();
          char[] ch1 = str1.ToCharArray();
          char [] ch3=new char[50];
          char [] ch4=new char[50];

          foreach (char c in ch)
          {
              if ((!(ch3.Contains(c)))&& (char.IsLetter(c)))
              {
                  ch3[j] = c;
                  j++;
              }

          }

          foreach (char c1 in ch1)
          {
```

```
            if ((!(ch4.Contains(c1))) && (char.IsLetter(c1)))
            {
                ch4[k] = c1;
                k++;

            }
        }
        Array.Resize(ref ch3, j);
        Array.Resize(ref ch4, k);

        foreach (char c3 in ch3)
        {
            if (ch4.Contains(c3))
            {
                count++;
            }
        }

        return count;
    }
}
```

Repeat Characters

Write a program to repeat the string multiple times provided with the below limitations.
a. Consider the index position of the input word starts with 1.
b. If the Input1 string length is odd, then the even index position characters should be removed from the input string and the remaining characters should be repeated based on Input2 value.
c. If the Input1 string length is even then the odd index position characters should be removed from the input string and the remaining characters should be repeated based on Input2 value.

Business Rules :
1. If the Input2 value is negative, then print “Invalid Input”.
2. If the Input2 value is greater than 10, then print “Input value is too long”.
3. If the length of Input1 is less than 2, then print “Input value is insufficient”.

Create a class named UserProgramCode that has the following static method
public static string repeatRemoveString(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:
Input consists of a string and an integer.
Output is a string.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
Price
4

Sample Output 1 :
PiePiePiePie

Sample Input 2 :
A
8

Sample Output 2 :
Input value is insufficient

```
using System;

using System.Collections.Generic;

using System.Linq;
```

```
using System.Text;


namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {


            string str = Console.ReadLine();
            int n = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine(Class1.repeatRemoveString(str,n));


        }
    }
}
 class Class1
    {
        public static string repeatRemoveString(string str, int n)
        {
            StringBuilder sb = new StringBuilder();
             StringBuilder sb1 = new StringBuilder();
```

```
int len = str.Length;
char [] ch=str.ToCharArray();
if (len % 2 != 0)
{
    for (int i = 0; i < len; i++)
    {

        if (i % 2 == 0)
        {
            sb.Append(ch[i]);

        }
    }


}

else  if (len % 2 != 0)
{
    for (int i = 0; i < len; i++)
    {
```

```
            if (i % 2 == 0)
            {
                sb.Append(ch[i]);


            }
        }



    }



    for (int k = 0; k < n; k++)
    {
        sb1.Append(sb);
    }


    return sb1.ToString();


  }
}
```

Find Span

Given an integer array as input, write a program to find the size of the largest Span in the given array,

Note: Span is the number of elements between two repeated numbers including both numbers.

Assume an array with single element has a span of 1.

Business rule:
If there is no number repeated in an array, print -1.
If there are two repeated integers in the input array, consider the first number and return the span.

Create a class named UserProgramCode that has the following static method
public static int getMaxSpan(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
The first line of the input consists of an integer, n that corresponds to the number of elements in the input array .
The next 'n' lines of input correspond to elements in the input array.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
5
1
2
1
1
3

Sample Output 1:
4

Sample Input 2 :
7
1
4
2
1
4
1
5

Sample Output 2 :
6

Check Anagrams

A anagram is a word or a phrase that can be created by rearranging the letters of another given word or phrase. We ignore white spaces and letter case. All letters of 'Desperation' can be rearranged to the phrase 'A Rope Ends It'.
Write a program to check whether the 2 given strings are anagrams or not.

Business Rules :
1. If there are any special characters (Space is not considered as special character) in either of the input strings, then store FALSE in the output variable.

Create a class named UserProgramCode that has the following static method
public static bool checkAnagram(string input1,string input2)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
Input consists of 2 strings.
Output is either "TRUE" or "FALSE".
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
tea
eat

Sample Output 1 :
TRUE

Sample Input 2 :
Desperation
A Rope Ends It

Sample Output 2 :
TRUE

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

namespace ConsoleApplication7
{
    class Class1
    {
        public static bool checkanagrams(string a, string b)
```

```
        {


            a.ToLower();
            b.ToLower();
            char[] c = a.ToCharArray();
            char[] d = b.ToCharArray();
            Array.Sort(c);
            Array.Sort(d);
            string e = new string(c);
            string f = new string(d);
            e=e.TrimStart();
            f = f.TrimStart();
            Console.WriteLine(e.Length);
            Console.WriteLine(f.Length);
            if (e.Length == f.Length)
            {

                for (int i=0;i<e.Length;i++)
                {
                    if (char.IsSymbol(e[i]) || char.IsSymbol(f[i]))
                    {
                        return false;
                    }

                }


                if (e == f)
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
            else
            {
                return false;
            }


        }

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

namespace ConsoleApplication7
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
            string a = Console.ReadLine();
            string b = Console.ReadLine();
            Console.WriteLine(Class1.checkanagrams(a, b));

        }
    }
}
```

Add and Reverse

Given an int array and a number 'k' as input, write a program to add all the elements in the array greater than the given number 'k'. Finally reverse the digits of the obtained sum and print it.

Include a class **UserProgramCode** with a static method "**addAndReverse**" that accepts 2 arguments and returns an integer.The first argument corresponds to the integer array and the second argument corresponds to the number k.

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

Example:
Input Array = {10,15,20,25,30,100}
Number = 15
sum = 20 + 25 + 30 + 100 = 175
output = 571

Input and Output Format:
The first line of the input consists of an integer that corresponds to the number of elements in the array.
The next n lines of the input consists of integers that correspond to the elements in the array.
The last line of the input consists of an integer that corresponds to the number, k.

Output consists of a single integer.

Sample Input
6
10
15
20
25
30
100
15

Sample Output

571

Count Digits

Write a method to find number of digits present in the given string.

Example:
Input1: Hell00 ho9 are u
Output1: 3

Include a class **UserProgramCode** with static method **countDigits** which accepts String value.

The return type should be int.

Create a class **Program** which would get the input and call the static method **countDigits** present in the**UserProgramCode**.
The input String consists only alphabets , numeric values and whitespaces (blank spaces).Otherwise display as "Invalid Input".

Input Output Format:
Input consists of a String.
Output consists of an integer which counts the number of digits in the given String.

Sample Input 1:
12345
Sample Output 1:
Number of digits present in given string are 5.

Sample Input 2:
hai12!
Sample Output 2:
Invalid Input

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;
```

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



            string str = Console.ReadLine();
            int d = Class1.countDigits(str);
            if (d == -1)
            {
                Console.WriteLine("Invalid Input");
            }
            else


                Console.WriteLine("Number of digits present in given string are "+d+".");
        }
    }
}
```

```
class Class1
  {
    public static int countDigits(string str)
    {

      int outp = 0;
      int count = 0;
      int count1 = 0;
      foreach (char c1 in str)
      {
        if(! (char.IsLetter(c1) || char.IsDigit(c1) || c1 == 32))
        {

          count1++;
        }
      }
      foreach (char c in str)
      {

        if (char.IsDigit(c))
        {
          count++;
```

```
            }


        }


        if (count1 == 0)
        {
            outp = count;
        }
        else
        {
            outp = -1;
        }
        return outp;
    }
}
```

Calculate Telephone Bill

Write a program which reads the number of calls as input and calculates the monthly telephone bills as per the following rules. Print the bill amount.

1. Minimum Rs. 200 for upto 300 calls.
2. Plus Rs. 0.60 per call for next 50 calls.
3. Plus Rs. 0.50 per call for next 50 calls.

4. Plus Rs. 0.40 per call for any call beyond 400 calls.

Example :

If the calls is 720, the calculation would be as follows :

1. First 300 calls the charge is Rs 200.

2. Next 50 the charge is 50 *.60=30.

3. Next 50 the charge is 50 *.50=25.

4. Balance calls (720-300-50-50) 320 the charge is 320 *.40=128.

5. Total charge = 200 + 30+25+128 = 383.

The calculated charge should be in double datatype which is rounded to 2 decimal places.

Include a class UserProgramCode with a static method calculateTelephoneBill which accepts an Integer. The return type (Double) should return the final bill amount.

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, which corresponds to number of calls.

Output consists of a Double (The final bill amount).

Refer sample output for formatting specifications.

Sample Input 1:

720

Sample Output 2:

383.00

Calculate Grade

Given an input as integer array with Student_ID and marks as the array element for multiple students in the format.
{Student_ID_1, Mark1, Student_ID_2, Mark2, Student_ID_3, Mark3, etc...}, write a program to calculate the grade of the student who has scored the maximum marks and print the output in the following format.

Student_ID XXX has passed in YYY

where XXX is the Student_ID and YYY is the grade.

1) If Mark is greater than or equal to 80, then store the grade as "DISTINCTION"
2) If Mark is less than 80 and greater than or equal to 60, then store the grade as "FIRST CLASS"
3) If Mark is less then to 60 and greater than or equal to 45 then store the grade as "SECOND CLASS"
4) If Mark is less than 45 and greater than or equal to 0, then store the grade as "FAIL".

Business rules:
1) If the Input contains any negative numbers, then print “Invalid Input”.
2) If the number of elements in Input array is less than or equal to 2, then print “Grading is not possible”.
3) If the number of elements in the Input array is odd, then print “Scores not provided for all Students”.

Create a class named UserProgramCode that has the following static method

public static string getGrade(int[] input1)
Create a class named Program that accepts the inputs and calls the static method

present in the UserProgramCode.

Input and Output Format:

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

The next 'n' lines of input consist of elements in the input array.

Refer business rules and sample output for formatting specifications.

Sample Input 1 :
10
101
80
102
75
103
50
104
60
100
40

Sample Output 1 :
Student_ID 101 has passed in DISTINCTION

Sample Input 2 :
4
22
9
-5
6

Sample Output 2 :

Invalid Input

```
using System;
```

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


namespace Calculate_grade
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = Convert.ToInt16(Console.ReadLine());
            int[] a = new int[n];
            for (int i = 0; i < n; i++)
            {
                a[i] = Convert.ToInt16(Console.ReadLine());
            }
            Console.WriteLine(UserProgramCode.grade(a));
        }
    }
}
using System;
using System.Collections.Generic;
```

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


namespace Calculate_grade
{
    class UserProgramCode
    {
        public static string grade(int[] a)
        {
            StringBuilder sb = new StringBuilder();
            string str = string.Empty;
            int res = 0;
            int max = 0;
            if (a.Length <= 2)
                return "Grading is not possible";
            else if (a.Length % 2 != 0)
                return "Scores not provided for all students";
            else
            {
                for (int i = 0; i < a.Length; i = i + 2)
                {
                    if (a[i] < 0 || a[i + 1] < 0)
```

```
            return "Invalid Input";
        a[1] = max;
        if (a[i + 1] > max)
        {
            max = a[i + 1];
            res = a[i];
        }
        if (max >= 80)
        {
            str = "DISTINCTION";
        }
        else if (max < 80 && max >= 60)
            str = "FIRST CLASS";
        else if (max < 60 && max >= 45)
            str = "SECOND CLASS";
        else if (max < 45 && max >= 0)
            str = "FAIL";
    }
}
sb.Append("Student_ID ");
sb.Append(res);
sb.Append(" has passed in ");
```

```
            sb.Append(str);

            string result = sb.ToString();

            return result;

        }

    }

}
```

All Vowels

Write a Program to check if given word contains exactly five vowels and the vowels are in alphabetical order. Assume there is no repetition of any vowel in the given string and all letters are in lower case.

Include a class **UserProgramCode** with a static method **testOrderVowels** which accepts a string and returns an integer. The method returns 1 if the condition stated above is satisfied. Else the method returns -1.

Create a Class **Program** which would be used to read a String and call the static method present in **UserProgramCode**.
If the method returns 1, print 'valid'. Else print 'invalid'.

Input and Output Format:

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

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

acebisouzz

Sample Output 1:

valid

Sample Input 2:

alphabet

Sample Output 2:

Invalid

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

namespace checkbatchcode
{
   class Program
   {
      static void Main(string[] args)
      {
         string str = Console.ReadLine();
         int result=usermaincode.testOrderVowels(str);
         if (result == -1)
         {
```

```
            Console.WriteLine("invalid");
        }
        else
        {
            Console.WriteLine("VAlid");
        }
        Console.ReadLine();
    }
  }
}

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

namespace checkbatchcode
{
  class usermaincode
  {
    public static int testOrderVowels(string str)
    {
```

```
int result = 0;
string str1 = "aeiou";
StringBuilder sb = new StringBuilder();
char[] ch = str.ToCharArray();
if (ch.Length > 100)
{
   result = -1;
   return result;
}
else
{
   foreach (char m in ch)
   {
      if (m == 'a' || m == 'e' || m == 'i' || m == 'o' || m == 'u')
      {
         sb.Append(m);
      }

   }

   string str2 = sb.ToString();
   if (str2 == str1)
```

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


  }
}
```

Digit Sum in String Array

Given a String array as input. Each element in this array may contain alphabets or digits. Develop code to add all the digits in every string and print the sum as an int. If two digits appear simultaneously do not consider it as one number. Ex- For 'Hyderabad 21' consider 2 and 1 as two digits instead of 21 as a number.

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

Example 1:
input1:
5
input2:

AAA1
B2B
4CCC
A5
ABCDE
output1:1+2+4+5=12
Example 2:
input1 :
3
input2 :
12
C23
5CR2

output1 : 1+2+2+3+5+2 = 15

Create a class named UserProgramCode that has the following static method

public static int sumOfDigits(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 of input consist of elements in the input string array.

Refer sample output for formatting specifications.

Sample Input :
5
AAA1
B2B
4CCC
A5
ABCDE

Sample Output :

12

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

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

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

namespace valid_voter
{
    class UserProgramCode
    {
        public static int eligible(string[] str)
        {
            int d,sum=0;
            StringBuilder sb=new StringBuilder();
            foreach (string a in str)
            {
                char[] ch = a.ToCharArray();
                for (int i = 0; i < ch.Length; i++)
                {
                      if (!char.IsLetterOrDigit(ch[i]))
                    {
                        return -1;
                    }
                      else if (char.IsDigit(ch[i]))
                    {
                        sb.Append(ch[i]);
                    }
```

```
            }
        }
        string s = sb.ToString();
        int y = Convert.ToInt32(s) ;
        while (y > 0)
        {
            d = y % 10;
            sum += d;
            y = y / 10;
        }
        return sum;


    }
  }
}
```

Sort the list

Write a program which reads an Integer(size of the list), a String List and a character, and to get the strings that will not start with the given character irrespective of case. Sort the elements in ascending order based on its length. Print the output list.
If the elements are having the same length, then display the elements in alphabetical order.

Include a class **UserProgramCode** with static method **GetTheElements** which accepts a String list and a character. The return type is List<String>.

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

In **GetTheElements** method

Only alphabets should be given in list , otherwise return "Invalid Input".
When the output list is empty, then return "List is Empty".
Otherwise return the appropriate result.

In **Program** class
Print the result which is return by **GetTheElements** method in **UserProgramCode.**

Input output format
The first line of the input is an integer that corresponds to n, the size of the list.

The next n lines of input correspond to the elements in the string list.
The line of the input contains the character.
The output is the List type List<String>.

Sample Input 1:

3
read
write
edit
e

Sample Output 1:

read
write

Sample Input2 :

2
Elegent
event
e

Sample Output2 :

List is Empty

Sample Input 3:

2
Eleg$ent
e^ent
e

Sample Output 3 :

Invalid Input

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;



namespace Sortlist

{
```

```
    class Program
    {
        static void Main(string[] args)
        {
            int n=Convert.ToInt16(Console.ReadLine());
            List<string> l =new List<string>(n);
            for (int i = 0; i < n; i++)
            {
                l.Add(Console.ReadLine());
            }
            char c = Convert.ToChar(Console.ReadLine());
            List<string> out1 = UserProgramCode.GetTheElements(l, c);
           foreach(string j in out1)
            {
                Console.WriteLine(j);
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace Sortlist
{
    class UserProgramCode
    {
        public static List<string> GetTheElements(List<string> l1, char c)
        {
            List<string> out1=new List<string>();
            List<string> out2=new List<string>();
            foreach (string i in l1)
            {
                string max = string.Empty;
                 if(i.StartsWith(c.ToString()))
                 {
                    char[] ch = i.ToCharArray();
                    foreach (char c1 in ch)
                    {
                        if (!char.IsLetter(c1))
                        {
                            out2.Add("Invalid Input");
                            return out2;
                        }
                    }
```

```
    }
    else
    {
        if (i.Length > max.Length)
        {
            max = i;
            out1.Add(i);
        }
        char[] ch = i.ToCharArray();
         foreach(char c1 in ch)
         {
             if (!char.IsLetter(c1))
             {
                out2.Add( "Invalid Input");
                 return out2;
             }
         }
    }
}
if (out1.Count == 0)
{
    out2.Add("List is empty");
    return out2;
```

```
        }
        else
            return out1;
    }
    }
}
```

Check Palindrome

Write a program to read a string input and to check that given string is a palindrome and contains at least two distinct vowels.

The vowels can be repetitive. Even if the same vowel occurs more than once, it should be considered as one vowel only. If the above condition is satisfied, print “Palindrome”, else print “Not Palindrome”.

Include a class UserProgramCode with a static method checkPalindrome which accepts a String. The return type (Integer) should return 1 if Palindrome, else -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, “Palindrome” or “Not Palindrome”.

Refer sample output for formatting specifications.

Sample Input 1:

himesh

Sample Output 1:

Not Palindrome

Sample Input 2:

ABEBA

Sample Output 2:

Palindrome

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

namespace cHECK_PALINDROME
{
    class Program
    {
        static void Main(string[] args)
        {
            string str = Console.ReadLine();
            int res =UserProgramCode.checkPalindrome(str);
            if (res == 1)
            {
                Console.WriteLine("Palindrome");
            }
            else
                Console.WriteLine("Not a palindrome");


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

namespace cHECK_PALINDROME
{
    class UserProgramCode
    {
        public static int checkPalindrome(string str)
        {
            int res = 0;
            StringBuilder sb=new StringBuilder();
            char[] ch = str.ToCharArray();
            char[] ch1 = str.ToCharArray();
```

```
            Array.Reverse(ch1);
            string sr = new string(ch1);
            string sr1 = new string(ch);
            char[] ch2 =sr1.ToCharArray();
            if (sr==sr1)
            {
                foreach (char i in ch2)
                {
                    if ((i == 'a') || (i == 'e') || (i == 'i') || (i == 'o') || (i ==
'u'))
                        sb.Append(i);
                }
                string s = sb.ToString();
                //Console.WriteLine(s);
                if (s.Length >= 2)
                {
                    int max = s.Length;
                    char[] c = s.ToCharArray();
                    int uni = c.Distinct().ToArray().Length;
                    //Console.WriteLine(uni);
                    if (uni<2)
                    {
                        res = -1;
                        return -1;
                    }
                    else
                        res = 1;
                    return res;

                }
                else
                {
                    return -1;
                }
            }
            else
            {
                res= -1;
                return -1;
            }


        }
    }
}
```

Next Highest Number

Write a progam that accepts an integer input and finds out all the combinations of the numbers possible with all the digits present in the input integer and then from the list of combinations,picks up the next higher number than the given input and prints it.

Business Rules :

1. If the given input integer is a negative number, then print -1 .
2. If the input contains more than 3 digits,then print -2.
3. If any of the digits present in input are found repetitive, then print -3 .

Create a class named UserProgramCode that has the following static method

public static int nextHighestNumber(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.

Sample Input 1:

376

Sample Output 1:

637

Sample Input 2:

-236

Sample Output 2:

-1

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

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

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

namespace next_highest_no
{
    class UserProgramCode
    {
        public static int nextHighestNumber(int input1)
        {
            int output = 0;
            int[] a = new int[3];
            int[] b = new int[3];
            int c2 = 0;
            int j = input1;
            int t = input1;
            int count = 0;
            while(input1!=0)
            {
                input1 = input1 / 10;
                count++;
            }
            if (count != 3)
                return -2;
            else if (input1 < 0)
                return -1;
            else
            {
                int i = 0;
                int n = 0;
                while (t!= 0)
                {
                    n = t % 10;
                    a[i] = n;
                    i++;
                    t = t / 10;
                }
                 if (a.Distinct().ToArray().Length != 3)
                {
                    //Console.WriteLine(a.Distinct().ToArray().Length);
                    return -3;
                }
                else
                {
                    for (int k = j + 1; k < 1000; k++)
                    {
                        Console.WriteLine(k);
                        int m = 0;
                        int l = 0;
                        int t1 = k;
                        c2 = 0;

                        while (t1 != 0)
                        {
```

```
                        m = t1 % 10;
                        if (a.Contains(m))
                        {
                            b[l] = m;
                            l++;
                            c2++;
                        }
                        else
                            break;
                        t1 = t1 / 10;
                    }
                    if (b.Distinct().ToArray().Length == 3 && c2==3)
                    {
                        output = b[2] * 100 + b[1] * 10 + b[0];
                        break;
                    }
                }
            }
        }

        //Console.WriteLine(output);
        return output;
    }
  }
}
```

Symmetric Difference

Given two integer arrays Input1 and Input2, write a program to calculate the Symmetric Difference of the two input arrays.

Symmetric difference is the difference of A Union B and A Intersection B
ie. [(A U B) - (A ^ B)] where A is the Input1 array and B is the Input2 array.
Union operation(U) merges the two arrays and makes sure that common elements appear only once. Intersection(^) operation includes common elements from both the arrays.
Finally sort the output and print the array.

Business Rules :

1. If any/all of the Input values in the Input1 array is negative, then print -1.
2. If any/all of the input values in the Input2 array is negative, then print -2.
3. If all the integers in Input1 array is common to Input2 array, then print -3.
4. If none of the integers in Input1 array is common to Input2 array, then print -3.

Example 1:
input:
5
Input1 :
11
5
14
26
3
input :
3
Input2 :
5
3
1

Output:
1
11
14
26
AUB: {11,5,14,26,3, 1} A^B: {5,3}
AUB - A^B = {1, 11, 14, 26}

Example 2:
Input :
3
Input1 :
2
16
-9
Input :
3
Input2 :
53
43
31
Output1:
-1

Create a class named UserProgramCode that has the following static method

public static int[] symmetricDifference(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, n that corresponds to the number of elements in the input1 array.

The next 'n' lines of input consist of elements in the input1 array.

The next line of the input consists of an integer, m that corresponds to the number of elements in the input2 array.

The next 'm' lines of input consist of elements in the input2 array.

Refer business rules and sample output for output formatting specifications.

Sample Input :

```
5
11
5
14
26
3
3
5
3

1
```

Sample Output :

```
1
11
14
26
```

Duplicate Characters

Write a Program which removes duplicate characters from the string. Your program should read a sentence (string) as input from user and return a string removing duplicate characters. Retain the first occurance of the duplicate character. Assume the characters are case – sensitive.

Include a class **UserProgramCode** with a static method **removeDuplicates** which accepts a string and returns a string.

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

Input and Output Format:

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

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

hi this is sample test

Sample Output 1:

hi tsample

Sample Input 2:

ABC DEF

Sample Output 2:

ABC DEF

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

namespace valid_voter
```

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

            string res = UserProgramCode.eligible(str);
            Console.WriteLine(res);
        }
    }
}

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

namespace valid_voter
{
    class UserProgramCode
    {
        public static string eligible(string str)
        {
           int p=0;
            StringBuilder sb = new StringBuilder();
            char[] ch = str.ToCharArray();
            for (int i = 0; i < ch.Length; i++)
            {
                for (int j = i + 1; j < ch.Length; j++)
                {
                    if (ch[i] == ch[j])
                    {
                        ch[j] = '!';
                        p =1;
                    }

                }
                    if (ch[i] != '!' && p==1)
                    {
                        sb.Append(ch[i]);
                    }

            }
            return sb.ToString();



        }
    }
}
```

ANOTHER:

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

```
namespace Remove_duplicates
{
    class Class1
    {
        public static string Removeduplicates(string s)
        {
            StringBuilder sb = new StringBuilder();
           // string[] str = s.Split(' ');
           // foreach (string i in str)
           // {
           //     char[] ch = i.ToCharArray();
           //     ch.Distinct().ToArray();
            //    sb.Append(i);
            //}
           // string s1 = sb.ToString();
//return s1;
            List<char> ch1=new List<char>();
            int flag = 0;
            string[] s1 = s.Split(' ');
            foreach (string s2 in s1)
            {
                flag = 0;
                char[] ch = s2.ToCharArray();
                foreach (char c in ch)
                {
                    if (ch1.Contains(c))
                    {
                    }
                    else
                    {
                        ch1.Add(c);
                        flag = 1;
                    }
                }
                if(flag!=0)
                ch1.Add(' ');
            }
            char[]ch2=ch1.ToArray();
            string str = new string(ch2);
            return str;


        }
    }
}
```

Validate Password

Write a method to validate given password. Apply following validations:

1. Minimum length should be 8 characters
2. Must contain any one of these three special characters @ or _ or #
3. May contain numbers or alphabets.
4. Should not start with special character or number
5. Should not end with special character

Include a UserProgramCode with a static method validatePassword.The method must return an integer 1 or -1. if it returns 1 then print a message "Valid Password". If the method returns -1 then print a message "Invalid Password".

Create a class Program which gets a string as input and calls the static method validatePassword present in the UserProgramCode.

Input and Output Format:
Input is a string which is the password.
Output is also a string which prints a message "Valid Password" or "Invalid Password".

Sample Input 1:
#bzdfh123c

Sample Output 1:
Invalid Password

Sample Input 2 :
jgu_123dfsd3

Sample Output 2:
Valid Password

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

namespace ConsoleApplication11
{
    class Program
    {
        static void Main(string[] args)
        {
            string a = Console.ReadLine();
            int res = Class1.validatepassword(a);
            if (res == 1)
            {
                Console.WriteLine("Valid Password");
```

```
            }
            else
            {
                Console.WriteLine("Invalid Password");
            }
        }
    }
}

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

namespace ConsoleApplication11
{
    class Class1
    {
        public static int validatepassword(string a)
        {
            int d = 0;
            char[] b=a.ToCharArray();
            if (b.Length > 7)
            {
                if(char.IsLetter(b[0]))
                {
                    if (char.IsLetter(b[b.Length-1]) || char.IsDigit(b[b.Length-1]))
                    {
                        for (int i = 0; i < b.Length; i++)
                        {
                            if (a[i] == '@' || a[i] == '_' || a[i] == '#')
                            {
                                d++;
                            }

                        }
                    }
                    else
                    {
                        return -1;
                    }
                }
                else
                {
                    return -1;
                }

            }
            else
            {
                return -1;
            }

            if (d > 0)
            {
                return 1;
            }
            else
```

```
            {
                return -1;
            }
        }
    }
}
```

Validate String

For a given String apply the following validations.

1. The given input String should be only four characters long. If not print -1.
2. First character can be an alphabet or digit. If not print -2 .
3. Second character must be uppercase alphabet. (eg 'M','R'.. any alpbabet A - Z). If not print -3 .
4. Third character must be a number and also between 5-9. If not print -4 .
If all the conditions are satisfied print 1.

Example 1:
input='vM7u3'
output = -1

Example 2:
input='&Mau'
output = -2

Example 3:
input='vrau'
output = -3

Example 4:
input='vR3a'
output = -4

Example 5:
input='vR5a'
output = 1

Create a class named UserProgramCode that has the following static method

public static int validateString(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 :

vR5a

Sample Output :

1

removeTens

Given an input array, write a program to remove all Tens present in the array and shift the other elements towards left and fill the trailing empty positions by 0 so that the modified array is of the same length of the given array a

Include a class **UserProgramCode** with static method removeTens that accepts an integer array and its size and returns the modified array.

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

Input and Output Format:

Input consists of an n+1 integers. The 1st integer corresponds to n, the size of the array. The remaining n integers correspond to the element in the array.

Output is the modified integer array.

Sample Input 1:

5

1
10
20
10
2

Sample Output 1:

1
20
2
0
0

Sample Input 2:

2
1
2

Sample Output 2:

1
2

```
 using System;

using System.Collections.Generic;
```

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

namespace remove_teens
{
    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 = userprogram.removeTens(n, a);
            foreach(int x in b)
            {
                Console.WriteLine(x);
            }
```

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

namespace remove_teens
{
    class userprogram
    {
        public static int[] removeTens(int n, int[] a)
        {
            int[] b = new int[n];
            int k = 0;
            for (int i = 0; i < n; i++)
            {
                if (!(a[i] == 10))
                {
                    b[k++] = a[i];
```

```
            }
        }
        for (int i = k + 1; i < n; i++)
        {
            b[k + 1] = 0;
        }
        return b;

    }
  }
}
```

Replace String

Write a program to form a new string by replacingeach character in the n'th word of the input string with given special character. Display resultant string in lowercase.

Example:

Input1: Hi are you fine Ram

Input2: 5

Input3: *

Output:hi are you fine ***

Business Rules:

1. Only alphabets should be given in input1 string Else return "-1" from the method and print “Invalid String” in Main.

2. Only positive number should be given for input2 Else return "-2" from the method and print “Number not positive” in Main.

3. Only special characters should be given for input3 Else return "-3" from the method and print “Character not a special character” in Main.

Include a class UserProgramCode with a static method replaceString which accept a String, an integer and a character. The return type (String) should return the Final String.

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

Input and Output Format:

Input consists of a String, an integer and a character, where String corresponds to the input string, the integer corresponds to the word number and the character values corresponds to the change character.

Output consists of a String.

Refer sample output for formatting specifications.

Sample Input 1:

Hi are you fine Ram

5

*

Sample Output 1:

hi are you fine ***

Sample Input 2:

Hi @re you fine Ram

5

*

Sample Output 2:

Invalid String

Sample Input 3:

Hi are you fine Ram

-5

*

Sample Output 3:

Number not positive

Sample Input 4:

Hi are you fine Ram

5

o

Sample Output 4:

Character not a special character

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

namespace replace_string
{
    class Program
    {
        static void Main(string[] args)
        {
            string s = Console.ReadLine();
            int n = int.Parse(Console.ReadLine());
            char ch = char.Parse(Console.ReadLine());
            string a = userprogram.replaceString(s, n, ch);
            if (a == "-1")
            {
                Console.WriteLine("Invalid String");
            }
```

```
            else if (a == "-2")
            {
                Console.WriteLine("Number not positive");
            }
            else if (a == "-3")
            {
                Console.WriteLine("Character not a special character");
            }
            else
            {
                Console.WriteLine(a);
            }

        }
    }

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

```
namespace replace_string
{
    class userprogram
    {
        public static string replaceString(string s, int n, char ch)
        {
            string[] str = s.Split();
            StringBuilder sb = new StringBuilder();
            StringBuilder sb1 = new StringBuilder();
            if (n < 0)
            {
                return "-2";
            }
            if ((char.IsLetter(ch)||char.IsDigit(ch)))
            {
                return "-3";
            }
            for (int i = 0; i < str.Length; i++)
            {
                char[] ch1 = str[i].ToCharArray();
                foreach (char x in ch1)
```

```
    {
        if (!(char.IsLetter(x)))
        {
            return "-1";
        }
    }
}
int l = str.Length;
if (n < l)
{
    for (int i = 0; i < str[n - 1].Length; i++)
    {
        sb.Append(ch);
    }
    str[n - 1] = sb.ToString();

    foreach (string y in str)
    {
        sb1.Append(y);
        sb1.Append(" ");
    }
    string a = sb1.ToString();
```

```
            return a;



        }

        return "null";




    }

  }

}
```

ANOTHER METHOD:

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

namespace replace_string
{
    class Program
    {
        static void Main(string[] args)
        {
           string w = Console.ReadLine();
            int h = int.Parse(Console.ReadLine());
            char h1 = char.Parse(Console.ReadLine());

            string a=Class1.replace(w, h,h1);

            if(a=="-1")
                Console.WriteLine("Invalid String");
            else  if (a == "-2")
                Console.WriteLine("Number not positive");
            else if (a == "-3")
                Console.WriteLine("Character not a special character");
            else
                    Console.WriteLine(a);

        }
    }
}
```

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

namespace replace_string
{
    class Class1
    {
        public static string replace(string w, int h, char h1)
        {

            StringBuilder sb = new StringBuilder();
            StringBuilder sb1 = new StringBuilder();
            string b = string.Empty;
            int a = 0;

            string[] str1 = w.Split(' ');
            foreach (string w1 in str1)
            {
                char[] w2 = w1.ToCharArray();
                foreach (char w3 in w2)
                {
                    if (!(char.IsLetter(w3)))
                    {
                        a=-1;
                       b= a.ToString();
                       return b;
                    }

                }
            }

            if (h < 0)
            {
                a = -2;
                b = a.ToString();
                return b;
            }

            if (char.IsLetterOrDigit(h1))
            {
                a = -3;
                b = a.ToString();
                return b;
            }

            string[] str = w.Split(' ');
            for (int i = 0; i < str[h - 1].Length; i++)
            {
                sb.Append("*");
            }

            str[h - 1] = sb.ToString();

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

```
                sb1.Append(str[i]+" ");

            }

            b = sb1.ToString();
            return b;

        }
    }
}
```

EMI Calculation

A person wants to apply for loan, but bank wants to check whether the person is current employee or retired or student based on given Date Of Birth (input1) and Loan EMI (input2).

1) The Bank can approve loan for student of Rs 200000 with rate of interest of 3% per Annum if age <= 22.
2) The Bank can approve loan for employee of Rs 300000 with rate of interest of 5% per Annum if age>22 and age=<45.
3) The Bank can approve loan for retired of Rs 500000 with rate of interest of 7% per Annum if age>45 and age<=100.

Write a program to calculate the EMI .

Note :

i) The Loan Amount has to be cleared in either 12/24/36/48 EMI months which is provided as input2.
ii) Round the output to the nearest integer if required.
iii)Age calculation to be done with respect to current date.

Business Rules:
i) If Date of Birth is not given in proper fomat(dd-MM-yyyy), then print -1.
ii) If EMI month is not 12 , 24 , 36 or 48 , then print -2.

Create a class named UserProgramCode that has the following static method

public static int checkEmpAgeEligible(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:

Input consists of a string that corresponds to Date of Birth and an integer that corresponds to EMI months.

Refer business rules and sample output for formatting specifications.

Sample Input 1 :

01-11-1983
12

Sample Output 1 :

26250

Sample Input 2 :
01/11/1983
48

Sample Output 2 :

-1

Sample Input 3 :

01-11-1983
48

Sample Output 3 :

7500

```
static void Main(string[] args)
        {
            string dob = Console.ReadLine();
            int ip = int.Parse(Console.ReadLine());
            int op = emi.emical(dob, ip);
             Console.WriteLine(op);
        }

public static int emical(string dob, int ip)
        {
            DateTime dt;
            int year = 0;
            float em = 0;
            int p = 0;
            bool res = DateTime.TryParseExact(dob, "dd-MM-yyyy", null,
System.Globalization.DateTimeStyles.None, out dt);
            if (res == false)
            {
                return -1;
            }
            if (!(ip == 12 || ip == 24 || ip == 36 || ip == 48))
            {
                return -2;
```

```
}
if (res == true)
{
    int month = DateTime.Now.Month - dt.Month;
    year = DateTime.Now.Year - dt.Year;
    if (month < 0)
    {
        year = year - 1;
        month = month + 12;
    }
}
if (year <= 22)
{
    em = (int)((200000+(200000 * .03)) / ip);
    Math.Round(em);
}
if (year > 22 && year <= 45)
{
    em = (int)((300000+(300000 * .05)) / ip);
    Math.Round(em);
}
if (year > 45 && year <= 100)
```

```
{
    em = (int)((500000+(500000 * .07)) / ip);
    Math.Round(em);
}
Console.WriteLine(em);
 p=Convert.ToInt32(em);
 return p;
```

Unique Counter

Write a program that reads a string and finds the number of unique characters in the string (ie the number of characters in the string that appear only once in the string). If the given string does not contain any unique characters print “No unique characters”.

Example -

Input: “HelloWorld”

Output: 5

Input: “coco”

Output: “No unique characters”

Include a class UserProgramCode with a static method uniqueCounter which accepts a String. The return type (Integer) should return the number of unique characters or -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 an Integer(number of unique characters) or a String (“No unique characters” if no unique characters are present).

Refer sample output for formatting specifications.

Sample Input 1:

HelloWorld

Sample Output 1:

5

Sample Input 2:

coco

Sample Output 2:

No unique characters

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;


namespace ConsoleApplication3
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            string in1 = Console.ReadLine();
            int op = UserProgramCode.replaceChar(in1);
            Console.WriteLine(op);
        }
    }
}


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

namespace ConsoleApplication3
{
    class UserProgramCode
    {
        public static int replaceChar(String a)
        {
            int Flag = 0;
            StringBuilder sb = new StringBuilder();
            char[] ch = a.ToCharArray();
```

```
            for (int i = 0; i < ch.Length; i++)
            {
                Flag = 0;
                for (int j = i + 1; j < ch.Length; j++)
                {
                    if (ch[i] == ch[j])
                    {
                        ch[j] = '!';
                        Flag = 1;
                    }
                    else
                        return -1;
                }
                if (Flag == 0&ch[i]!='!')
                {
                    sb.Append(ch[i]);
                }
            }
            string klp = sb.ToString();
            int e = klp.Length;
            return e;


        }
    }
}
```

Reverse the adjacent pairs of letters

Write a program to swap the adjacent letters from the given string. If the string has an odd number of letters, the last letter is unchanged.

Include a class **UserProgramCode** with static method **swapPairs** that accepts the string and return type should be string.

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

If the input contains special characters or numbers, display "Invalid Input" in **swapPairs**() otherwise display the resultant string.

Input Output format:
The input and output should be a String.

Sample input 1:
Newyork
Sample output 1:
eNywrok

Sample input 2:
New!@
Sample output 2:
Invalid Input

```
 class Program
   {
       static void Main(string[] args)
       {
           string input = Console.ReadLine();
           string p = Usermaincode.swap(input);
           Console.WriteLine(p);
       }
   }
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication3
{
    class Usermaincode
   {
       public static string swap(string input)
       {
           StringBuilder sb = new StringBuilder();
```

```
    char temp = ' ';
    int l = input.Length;
    char[] ch = input.ToCharArray();
    foreach (char u in ch)
    {
        if (char.IsLetter(u))
        {
        }
        else
            return "Invalid Input";
    }

if (l % 2 == 0)
    {

        char[] ch2 = input.ToCharArray();

        for (int i = 0; i < l - 1; i += 2)
        {
            temp = ch2[i];
            ch2[i] = ch2[i + 1];
            ch2[i + 1] = temp;
```

```
    }

    string o=new string(ch2);

    return o;
}
else
{
    string r = input.Substring(0, l - 1);
    char[] ch1 = r.ToCharArray();
    for (int i = 0; i < r.Length - 1; i += 2)
    {
        temp = ch1[i];
        ch1[i] = ch1[i + 1];
        ch1[i + 1] = temp;
    }
    string t = new string(ch1);
    sb.Append(t);
    sb.Append(input.Substring(l - 1, 1));

    string ottp = sb.ToString();
    return ottp;
```

```
        }
      }
    }
}
```

Interleaved Words

Given three input strings input1,input2 and input3, write a program to check whether input3 string is an interleaved word of input1 and input2 strings (i.e. input3 = concatenation of input1 and input2 strings,such that input2 string is concatenated at the end of the input1 string eg. if 'game' is input1 and 'center' is input2,input3 should be equivalent to 'gamecenter' and not 'centergame').
If it's a interleaved word, then print the following :
input3 is a interleaved word of input1 and input2 together

Replace input1, input2 and input3 with the corresponding inputs. Ignore case sensitiveness in input and use lowercase to print the output.

Business rule:
1) If input1 or input2 strings contains any number, then print -1.
2) If both input1 and input2 strings are same, then print -2.
3) If input1 or input2 contains any special characters, then print -3.

Create a class named UserProgramCode that has the following static method

public static string checkInterleavedword(string input1,string input2,string 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 strings – input1, input2 and input 3.

Refer business rules and sample output for output formatting specifications.

Sample Input 1 :
foreign

land
foreignland

Sample Output 1 :

foreignland is a interleaved word of foreign and land together

Sample Input 2 :
string1
set
string1set

Sample Output 2 :

-1

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;


namespace ConsoleApplication3

{

    class Program

    {

        static void Main(string[] args)

        {

            string in1 = Console.ReadLine();

            string in2 = Console.ReadLine();

            string in3 = Console.ReadLine();
```

```
            string op = UserProgramCode.replaceChar(in1, in2,in3);
            Console.WriteLine(op);
        }
    }
}

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

namespace ConsoleApplication3
{
    class UserProgramCode
    {
        public static string replaceChar(String a, String b,string c)
        {
            string s = "1234567890";
            string p = "!@#$%^&*()";

            StringBuilder sb = new StringBuilder();
            sb.Append(a + b);
            string q = sb.ToString();
            foreach (char y in q)
            {
                if (s.Contains(y))
```

```
                {
                    return "-1";
                }
                else if (p.Contains(y))
                {
                    return "-3";
                }
            }
            if (a == b)
            {
                return "-2";
            }
            string w="";
            if(q==c)
            {
                w=c+ " is a interleaved word of "+a+ " and "+ b +" together";

            }
            return w;
        }
    }
}
```

String Occurences

Write a program to count the number of occurences of second word of second sentence in the first sentence.

Return the count as output.
Note - Consider case.
Include a class **UserProgramCode** with a static method **countNoOfWords** which accepts two string variables. The return type is the integer.

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

Input and Output Format:

Input consists of two strings with maximum size of 100 characters.

Output consists of an integer.

Refer sample output for formatting specifications.

Sample Input 1:

abc bcd abc bcd abc abc

av abc

Sample Output 1:

4

Sample Input 2:

ABC xyz AAA

w abc

Sample Output 2:

0

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

namespace ConsoleApplication3
{
    class Program
    {
        static void Main(string[] args)
        {
            string in1 = Console.ReadLine();
            string in2 = Console.ReadLine();
            int op = UserProgramCode.replaceChar(in1, in2);
            Console.WriteLine(op);
        }
    }
}

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

```
namespace ConsoleApplication3
{
    class UserProgramCode
    {
        public static int replaceChar(String a, String b)
        {
            int c=0;
            string[] q = a.Split();
            string[] w = b.Split();
            foreach (string e in q)
            {
                if (e == w[1])
                {
                    c++;
                }
            }
            return c;



        }
    }
}
```

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 array.

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.Threading.Tasks;


namespace repint
{
    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());
            userprogram.findRepeatedIntegers(n, a);
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;


namespace repint
{
    class userprogram
    {
        public static void findRepeatedIntegers(int n, int[] a)
        {
            int count = 0, count1 = 0;
            List<int> l = new List<int>();
            for (int i = 0; i < n; i++)
            {
                if (a[i] < 0)
                {
                    count++;


                }
            }
            if (count > 0)
            {
                Console.WriteLine("Array contains negative numbers");
            }
```

```
else
{

    for (int i = 0; i < n - 1; i++)
    {
        for (int j = i+1; j < n; j++)
        {
            if (a[i] == a[j])
            {
                l.Add(a[j]);
                count1++;
            }
        }
    }
    if (count1 == 0)
    {
        Console.WriteLine("No repeated numbers");
    }
    else
    {
        List<int> s = l.Distinct().ToList();
        s.Sort();
```

```
                foreach (int k in s)
                    Console.WriteLine(k);
            }
        }
    }
}
```

Password Encryption

There is a online shopping site which stores their customer userid and password in their database. However, the password is encrypted before storing them into the DB for maintaining the security.

Consider the password and key used for encryption are given as input1 and input2 respectively.Write a function to encrypt based on encryption logic as follows
1) Replace the first letter of every word in input1 with input2.
2) If first letter of any word of input1 matches with input2 then replace the first letter of that word of input1 with the next alphabet and add # symbol next to it.

Business Rules :
1)If the password given is empty, then assign Invalid Password to the output1 variable.
2)If the password given consists of digits or special characters,then assign Invalid Input to the output1 variable

Create a class named UserProgramCode that has the following static method

public static string replaceChar(String input1,String input2)

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

Input and Output Format:
Input consists of 2 strings. The first string corresponds to the password and the second string corresponds to the key used.

Output consists of a string.

Refer sample output and business rule for output formatting specifications.

Sample Input 1 :
Red Green
A

Sample Output 1 :
Aed Areen

Sample Input 2 :
Red Sed Yellow

R

Sample Output 2 :

S#ed Red Rellow

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;


namespace ConsoleApplication3
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            string in1 = Console.ReadLine();
            string in2 = Console.ReadLine();
            string op = UserProgramCode.replaceChar(in1, in2);
            Console.WriteLine(op);
        }
    }
}

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

namespace ConsoleApplication3
{
    class UserProgramCode
    {
        public static string replaceChar(String a, String b)
        {
            StringBuilder sb1=new StringBuilder();
            string h="";
```

```
string u = "1234567890!@#$%^&*()";
if (b == h)
{
    sb1.Append("Invalid password");
    return sb1.ToString();
}
char nv=Convert.ToChar(b);
if (!char.IsDigit(nv)||char.IsDigit(nv))
{
    sb1.Append("Invalid");
    return sb1.ToString();
}



StringBuilder sb=new StringBuilder();
string[] r = a.Split(' ');
foreach (string s in r)
{
    int t = s.Length;
    char[] ch = s.ToCharArray();
    //char[] t = b.ToCharArray();
    if (s.Substring(0, 1) == b)
    {
        char q = (char)(ch[0] + 1);
       string n= s.Substring(1, t - 1);
        sb.Append(q);
```

```
                    sb.Append('#');
                    sb.Append(n);
                    sb.Append(' ');
                }
                else if (s.Substring(0, 1) != b)
                {
                    string p = s.Substring(1, t-1);
                    sb.Append(b);
                    sb.Append(p);
                    sb.Append(' ');
                }
            }
            return sb.ToString();
        }
    }
}
```

Convert Roman to Decimal

Write a program to convert the given roman number to decimal number.

Example :
Input string: XVII
Output variable:10+5+1+1 = 17

The input string should contain only the alphabets given below (in upper case) .

Valid alphabets are I, V, X, L, C, D, and M
'I': The corresponding value = 1
'V': The corresponding value = 5
'X': The corresponding value =10
'L': The corresponding value = 50
'C': The corresponding value = 100
'D': The corresponding value = 500
'M': The corresponding value =1000

Include a class UserProgramCode with a static method **convertRomanToDecimal** that accepts a string and returns an integer. The method returns -1 if the input string is not valid.

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

Input and Output Format:

Input consists of a string.

Output consists of an integer.

Sample Input 1:
XII

Sample Output 1:
12

Sample Input 2:
DCL

Sample Output 2:
650

Sample Input 3:
DCA

Sample Output 3:
-1

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;


namespace ConsoleApplication3

{

    class Program

    {

        static void Main(string[] args)

        {

            string in1 = Console.ReadLine();

            int op = UserProgramCode.replaceChar(in1);

            Console.WriteLine(op);

        }
```

```
    }
}


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


namespace ConsoleApplication3
{
    class UserProgramCode
    {
        public static int replaceChar(String s)
        {
      int value=0;
       foreach (char s1 in s)
        {
            if (s1 != 'I' && s1 != 'V' && s1 != 'X' && s1 != 'L' && s1 != 'C' && s1 != 'D' && s1 != 'M')
            {
                return -1;
            }


            if (s1.Equals('I'))
                value += 1;
            else if(s1.Equals('V'))
```

```
                value += 5;
            else if (s1.Equals('X'))
                value += 10;
            else if (s1.Equals('L'))
                value += 50;
            else if (s1.Equals('C'))
                value += 100;
            else if (s1.Equals('D'))
                value += 500;
            else if (s1.Equals('M'))
                value += 1000;

        }
        return value;



        }
    }
}
```

Get Common Elements

Write a program that accepts two lists and finds out the elements that are common in both of the given input lists and consolidates the common elements in another list.

Business Rule:

Only positive number should be given in input List. Else print -1 in the output.

Create a class named UserProgramCode that has the following static method

public static List<int> getCommonItems(List<int> input1, List<int> input2)

Create a class named Program that accepts the 2 input lists 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 m, the number of elements in the first list. The next 'm' lines consist of elements in the first list.

The next line of the input consists of an integer that corresponds to n, the number of elements in the second list. The next 'n' lines consist of elements in the second list.

Output consists of the list that contains the common elements in both the lists. If any of input list elements are negative, print -1.

Sample Input :

```
5
5
6
7
8
1
4
5
2
3
1
```

Sample Output :

```
5
1
```

Calculate Discount

Given two input integer arrays input1 and input2 which contains the details of users booking cars online in the format of key value pairs.
input1 (Key,Value) = (UserId1,Bookingmonth1,UserId2,Bookingmonth2,...so on)
input2 (Key,Value) = (UserId1,BookingAmount1,UserId2,BookingAmount2,...so on)
Write a Program to calculate the discount amount based on the Booking amount and Booking month for every UserId by considering the respective discount rate offered to them and store the result in an output integer array .

The discount amount should be calculated using the following specifications

Discount Rate	Booking Amount	
	>=500000	>=100000 and < 500000
Discount Rate for Jan-April	25	15
Discount Rate for May-August	20	10
Discount Rate for Sep-December	15	5

Output should be in the below format:
output1 (Key,Value) = (UserId1,DiscountAmount1,UserId2,DiscountAmount2,...so on)

Note:
1)Formula: Discount Amount = Booking Amount * (Discount Rate/100)

BUSINESS RULE:
1.If Booking Amount is less than 100000, then discount amount should be zero.
2.If any of the booking month is invalid, then print -1.
3.If any of the booking amount is negative, then print -2.
4.If any of the Booking Month or Booking Amount is missing for any UserId, then print -3 .

Create a class named UserProgramCode that has the following static method

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

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

Input and Output Format:

The first line of the input consists of an integer, m that corresponds to the number of elements in the input array 1 .

The next 'm' lines of input correspond to elements in the input array 1.

The next line of the input consists of an integer, n that corresponds to the number of elements in the input array 2 .

The next 'n' lines of input correspond to elements in the input array 2.

Refer business rules and sample output for formatting specifications.

Sample Input 1 :

```
8
1010
11
1011
02
1012
07
1013
09
8
1010
700000
1011
300000
1012
150000
1013
100000
```

Sample Output 1 :

```
1010
105000
1011
45000
1012
15000
1013
5000
```

Sample Input 2 :

8
1010
11
1011
02
1012
07
1013
09
6
1010
700000
1011
300000
1012
100000

Sample Output 2:

-3

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

namespace valid_voter
{
    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 m = int.Parse(Console.ReadLine());
            int[] k = new int[m];
            for (int j = 0; j < m; j++)
            {
                k[j] = int.Parse(Console.ReadLine());
            }
            int[] q = UserProgramCode.eligible(a, k);
            foreach (int i in q)
            {
                Console.WriteLine(i);
            }
        }
    }
```

```
}

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

namespace valid_voter
{
    class UserProgramCode
    {
        public static int[] eligible(int[] a,int[]b)
        {
            int[] u=new int[1];
            if (a.Length != b.Length)
            {
                u[0] = -3;
                return u;
            }

            for (int i = 1; i < a.Length; i=i+2)
            {
                if (a[i] < 1 || a[i] > 12)
                {
                    u[0] = -1;
                    return u;
                }
                if (b[i] < 0)
                {
                    u[0] = -2;
                    return u;
                }
            }
            if (a.Length != b.Length)
            {
                u[0] = -3;
                return u;
            }


            int n=a.Length;
            int[] r = new int[n];
            for (int i = 1; i < n; i = i + 2)
            {
                if (a[i] >= 1 && a[i] <= 4)
                {
                    if (b[i] > 500000)
                    {
                        r[i] = (b[i] * 25 / 100);
                    }
                    else if(b[i]>=100000 && b[i]< 500000)
                    {
                        r[i] = (b[i] * 15 / 100);
                    }
                    else
                    {
                        r[i] = 0;
                    }
```

```
            }
            if (a[i] >= 5 && a[i] <= 8)
            {
                if (b[i] > 500000)
                {
                    r[i] = (b[i] * 20 / 100);
                }
                else if (b[i] >= 100000 && b[i] < 500000)
                {
                    r[i] = (b[i] * 10 / 100);
                }
                else
                {
                    r[i] = 0;
                }
            }
            if (a[i] >= 9 && a[i] <= 12)
            {
                if (b[i] > 500000)
                {
                    r[i] = (b[i] * 15 / 100);
                }
                else if (b[i] >= 100000 && b[i] < 500000)
                {
                    r[i] = (b[i] * 5 / 100);
                }
                else
                {
                    r[i] = 0;
                }
            }
        }
        for (int i = 0; i < n; i = i + 2)
        {
            r[i] = a[i];
        }
        return r;




    }
}
}
```

Insurance Guide

An Insurance company follows the following rules to calculate premium.

(1) If a person's health is excellent and the person's age is in the range [25,35] (both 25 and 35 inclusive) and lives in a city and is a male then the premium is 4 Rs. per thousand and his policy amount cannot exceed Rs. 2 lakhs rupees.

(2) If a person satisfies all the above conditions except that the sex is female then the premium is 3 Rs. per thousand and her policy amount cannot exceed Rs. 1 lakh rupees.

(3) If a person's health is poor and the person's age is in the range [25,35] (both 25 and 35 inclusive) and lives in a village and is a male then the premium is 6 Rs. per thousand and his policy cannot exceed Rs. 10,000 rupees.

(4) In all other cases the person cannot be insured.

Write a program to display premium and maximum policy amount for given inputs.

Input1 - Health condition ('E' for excellent and 'P' for poor health)

input2 - Age

input3 - Gender('F' for female,'M' for Male)

input4 - Location('C' for City,'V' for Village)

Output1 - Premium per Thousand

Output2 - Maximum Insurance Amount the person can avail.

Example:

input1: E

input2: 30

input3: F

input4: C

Output1: 3

Outpu2: 100000

Business Rule:

1. If the person can't be insured then print “The person cannot be insured”.

2. If the person's age is more than 60 then print “Age limit Exceeded”.

Include a class UserProgramCode with a static method InsuranceGuide which accept three characters and an integer. The return type is void.

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

Input and Output Format:

Input consists of a character, which corresponds to the health condition, an Integer, which corresponds to the age, a character, which corresponds to the gender, a character, which corresponds to the location.

Refer sample output for formatting specifications.

Sample Input 1:

E

30

F

C

Sample Output 1:

3

100000

Sample Input 2:

E

70

F

C

Sample Output 2:

Age limit Exceeded

Sample Input 3:

P

50

F

V

Sample Output 3:

The person cannot be insured

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

namespace ConsoleApplication18
{
    class Program
    {
        static void Main(string[] args)
        {
            char a = char.Parse(Console.ReadLine());
            int n = int.Parse(Console.ReadLine());
            char b = char.Parse(Console.ReadLine());
            char c = char.Parse(Console.ReadLine());
            userprogramcode.yo(a,n,b,c);


        }
    }
}

namespace ConsoleApplication18
{
    class userprogramcode
    {
        public static void yo(char a, int n, char b, char c)
        {
            int k;
            int l;
            if (n <= 60)
            {

                if (n >= 25 & n <= 35)
                {
                    if (a == 'E' & b == 'M' & c == 'C')
                    {
                        k = 4;
                        l = 20000;
                        Console.WriteLine(k);
                        Console.WriteLine(l);
                    }
                    else if (a == 'E' & b == 'F' & c == 'C')
                    {
                        k = 3;
                        l = 100000;
                        Console.WriteLine(k);
                        Console.WriteLine(l);
                    }
                    else if (a == 'P' & b == 'M' & c == 'V')
                    {
                        k = 6;
                        l = 10000;
```

```
                    Console.WriteLine(k);
                    Console.WriteLine(l);
                }

                else
                {
                    Console.WriteLine("The person cannot be insured");
                }

            }
            else
            {
                Console.WriteLine("The person cannot be insured");
            }

        }
        else
        {
            Console.WriteLine("Age limit Exceeded");
        }

    }
    }
}
```

Calculate New Salary

The HR of an IT company fixes the salary of a new joinee who joins the company with the previous experience. Given the year of experience, Technology expertise and the previous salary drawn, the HR fixes the current salary as follows:
1. Straight 30% hike from the previous salary drawn.
2. a. Add Extra 5% if the year of experience is more than 3 and less than or equal to 5 years.
b. Add Extra 10% if the year of experience is more than 5 and less than or equal to 8 years.
c. Add Extra 15% if the year of experience is more than 8 years
3. Technology expertise is classified broadly into two namely common skills (CS) and rare skills (RS) People having rare skills get an extra 5% increase from their previous salary drawn.

Write a program to calculate the salary the HR fixes for the new joinee given the experience (input1), technology expertise classification (input2) and the previous salary drawn (input3) and print the output in the given format

Your Salary is fixed as Rs YYYYYY
where YYYYYY is the calculated new salary. (The digits depends upon the salary calculated.)
Salary is rounded off and displayed as an integer.

Business rules:
1. If the experience is given more than 25 or less than 0, then print “Invalid Experience”
2. If the technology expertise classification is given other than CS or RS, then print “Invalid Technology expertise classification”
3. If the previous salary is given more than 100000 or less than 0, then print “Invalid Salary”

Create a class named UserProgramCode that has the following static method

public static int calculateNewSalary(int,string,int)

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 experience (input 1).

The second line of the input consists of a string that corresponds to technology expertise (input 2).

The third line of the input consists of an integer that corresponds to previous salary drawn (input 3).

Refer business rules and sample output for formatting specifications.

Sample Input 1 :
7
RS
30000

Sample Output 1 :

Your Salary is fixed as Rs 43500

Sample Input 2 :

7
RSS
30000

Sample Output 2 :

Invalid Technology expertise classification

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

namespace valid_voter
{
    class Program
    {
        static void Main(string[] args)
        {
            int m = int.Parse(Console.ReadLine());
            string s = Console.ReadLine();
            int a = int.Parse(Console.ReadLine());
            int b = UserProgramCode.eligible(m, s, a);
            if (b == -1)
            {
                Console.WriteLine("Invalid Experience");
            }
            else if (b == -2)
            {
                Console.WriteLine("Invalid Technology expertise classification");
            }
            else if (b == -3)
            {
                Console.WriteLine("Invalid Salary");
            }
            else
            {
                Console.WriteLine("Your Salary is fixed as Rs {0}", b);
            }
        }
    }
}

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

namespace valid_voter
{
    class UserProgramCode
    {
        public static int eligible(int a, string s,int b)
        {
            int z=0,c=0;
```

```
            if (a < 0 || a > 25)
            {
                return -1;
            }
            if (s!="CS" && s!="RS")
            {
                return -2;
            }
            if (b < 0 || b > 100000)
            {
                return -3;
            }
            if (s == "CS")
            {
                if (a > 3 && a <= 5)
                {
                    z = b * 35 / 100;
                    c = b + z;

                }
                else if (a > 5 && a <= 8)
                {
                    z = b * 40 / 100;
                    c = b + z;
                }
                else if (a > 8)
                {
                    z = b * 45 / 100;
                    c = b + z;
                }

            }
            else if(s=="RS")
            {
                if (a > 3 && a <= 5)
                {
                    z = b * 40 / 100;
                    c = b + z;

                }
                else if (a > 5 && a <= 8)
                {
                    z = b * 45 / 100;
                    c = b + z;
                }
                else if (a > 8)
                {
                    z = b * 50 / 100;
                    c = b + z;
                }

            }
            return c;
        }
    }
}
```

Test Vowels Order

Write a program that takes a string as input and checks whether the given string contains exactly five vowels and the vowels should be in alphabetical order. Assume there is no repetition of any vowel in the given string.

If the above condition is satisfied, print 1.
If the above condition is not satisfied, print 2.

Business rule:

If there is a repetition of vowels in the given string, print -1.

Create a class named UserProgramCode that has the following static method

public static int testOrderVowels (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 formatting specifications.

Sample Input :

acebisouzz

Sample Output :

1

```
using System;
```

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

namespace ConsoleApplication1
{
    class Usermaincode
    {
        public static int testOrderVowels(string input1)
        {
            int count = 0;
            StringBuilder sb=new StringBuilder();

            char[] ch = input1.ToCharArray();
            foreach (char i in ch)
            {
                if (char.IsLetter(i))
                {
                    if (i == 'a' || i == 'e' || i == 'i' || i == 'o' || i == 'u')
                    {
                        sb.Append(i);
                        count++;
                    }
```

```
    }
}

string vow = sb.ToString();
Console.WriteLine(vow);
char[] h = vow.ToCharArray();
// h = h.Distinct().ToArray();
string t = new string(h.Distinct().ToArray());



if (count == 5)
{
    if (vow == "aeiou")
        return 1;
    else
    {
        if (t.Length < 5)
            return -1;
        else
            return 2;
    }
}
```

```
            else
            {
                if (count > 5)
                    return -1;


            }
            return 0;


        }
    }
}


class Program
    {
        static void Main(string[] args)
        {
            string input1 = Console.ReadLine();
            int e = Usermaincode.testOrderVowels(input1);
            Console.WriteLine(e);


        }
    }
```

Calculate Charge

A parking garage charges a Rs.20 minimum fee to park for up to three hours. The garage charges an additional Rs. 5 per hour for each hour or part thereof in excess of three hours. The maximum charge for any given 24-hour period is Rs.100. Assume that no car parks for longer than 24 hours at a time.

Write a program which accepts two DateTime inputs as string datatype. Convert the inputs to DateTime DataType and calculate the parking charge for the vehicle.

Validations:

1. DateTime String format is “yyyy-MM-dd:HH:mm:ss” eg: 2009-10-21:14:35:45 .

Return -1 as error code for other formats. The first parameter in the method will refer to the checkinDate and the second would refer to checkoutDate.

2. CheckoutDateTime should be greater than check in time. Return -2 if that is not the case.

3. If the duration exceeds 24 hrs return -3 as error code.

Include a class UserProgramCode with a static method calculateCharge which accepts two Strings. The return type (Integer) should return the parking charge. Also follow the validations.

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

Input and Output Format:

Input consists of two Strings, the first String corresponds to the CheckinDateTime and the second String corresponds to the CheckoutDateTime.

Output consists of an Integer (the parking charges) or, a String “Invalid Date format” if -1 is returned, “CheckoutDateTime is less than CheckinDateTime” if -2 is returned, “Duration exceeds 24 hrs” if -3 is returned.

Refer sample output for formatting specifications.

Sample Input 1:

2009-10-21:14:35:45

2009-10-21:16:35:45

Sample Output 1:

20

Sample Input 2:

2009-10-211:14:35:45

2009-10-21:16:35:45

Sample Output 2:

Invalid Date format

Sample Input 3:

2009-10-21:14:35:45

2009-10-21:10:35:45

Sample Output 3:

CheckoutDateTime is less than CheckinDateTime

Sample Input 4:

2009-10-20:14:35:45

2009-10-21:16:35:45

Sample Output 4:

Duration exceeds 24 hrs

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


namespace calculate_charge
{
   class Program
   {
      static void Main(string[] args)
      {
         string input1 = Console.ReadLine();
         string input2 = Console.ReadLine();
         int res =Class1.calculatecharges(input1,input2);


         if (res == -1)
         {
            Console.WriteLine("Invalid Date format");
```

```
            }
            else if (res == -2)
            {
                Console.WriteLine("CheckoutDateTime is less than CheckinDateTime");
            }
            else if (res == -3)
            {
                Console.WriteLine("Duration exceeds 24 hrs");
            }
            else
            {
                Console.WriteLine(res);


            }



        }
    }
}
```

```
using System;
```

```
using System.Collections.Generic;

using System.Linq;

using System.Text;


namespace calculate_charge

{

   class Class1

   {

      public static int calculatecharges(string input1, string input2)

      {

         DateTime dt;

         DateTime dt1;

         double output = 0;

         double duration = 0;


         bool res = DateTime.TryParseExact(input1, "yyyy-MM-dd HH:mm:ss", null,
System.Globalization.DateTimeStyles.None, out dt);

         bool res1 = DateTime.TryParseExact(input2,"yyyy-MM-dd HH:mm:ss", null,
System.Globalization.DateTimeStyles.None, out dt1);


         if (res == false || res1 == false)

         {

            return -1;

         }
```

```
else if (dt > dt1)
{
    return -2;
}

    duration = Math.Ceiling(dt1.Subtract(dt).TotalHours);
    if (duration > 24)
    {
        return -3;
    }

    else if (duration <= 3)
    {
        output = 20;
    }
    else if (duration > 3 && duration <= 24)
    {
        output = 20 + 5 * (duration-3);
    }
    else
    {
```

```
            }


            if (output > 100)
            {
                output = 100;
            }


          return (int)output;


       }


    }
}
```

Permutations

Given a String as input, write a program to find all possible permutations of the given input. If the string contains duplicate characters, remove the duplicate characters. Sort the array alphabetically and print the resultant array. The strings in input and output are case sensitive.

Business Rule:
1. If string contains any special characters or any digits as input, then print “Invalid Input” .

Create a class named UserProgramCode that has the following static method
public static List<string> permString(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.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
cat

Sample Output 1 :
act
atc
cat
cta
tac
tca

Sample Input 2 :
ffin

Sample Output 2 :
fin
fni
ifn
inf
nfi
nif

Sample Input 3 :
ffin%2

Sample Output 3 :
Invalid Input

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string set = Console.ReadLine();
            List<string> output = UserProgramCode.FindPermutations(set);
            foreach (var s in output)
            {
```

```
                Console.WriteLine(s);
            }
            Console.ReadKey();

        }
    }
}

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static List<string> FindPermutations(string set)
        {
            var output = new List<string>();
            if (set.Length == 1)
            {
                output.Add(set);
            }
            else
            {
                var chars = set.ToCharArray();
                foreach (var c in chars)
                {
                    var tail = chars.Except(new List<char>() { c });
                    foreach (var tailPerms in FindPermutations(new
string(tail.ToArray())))
                    {
                        output.Add(c + tailPerms);
                    }
                }
            }
            return output;
        }
    }
}
```

Day of Week

Write a program to find out the day of week for given input date which is in string format (MM-dd-yyyy). The output should be in titlecase.

Include a class UserProgramCode with a static method **getDay** which accepts a string . 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.

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

Refer sample output for formatting specifications.

Sample Input 1:

07-13-2012

Sample Output 1:

Friday

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string set = Console.ReadLine();
            string a = UserProgramCode.FindPermutations(set);
            Console.WriteLine(a);

        }
    }
}
```

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static string FindPermutations(string set)
        {
            string format = "MM-dd-yyyy";
            DateTime dt;
            DateTime.TryParseExact(set, format, null,
System.Globalization.DateTimeStyles.None, out dt);
            return dt.DayOfWeek.ToString();

        }
    }
}
```

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

Count the number of Occurrences

Write code to read two lines(String) and count the number of occurrences of second word of second sentence in the first sentence. Print the count.

Note - Consider case.

Example:
Input1: Hi this is cognizant Academy
Input2: Hello this is a trainee
Output: 1

Business Rule:
If there is no occurrence of the second word of second sentence with respect to the first sentence, return 0.

Include a class UserProgramCode with a static method countNoOfWords which accepts two strings. The return type (integer) should return the count.

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 an integer (the count).
Refer sample output for formatting specifications.
Sample Input 1:
Hi this is cognizant Academy
Hello this is a trainee
Sample Output 1:
1

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

```
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string set = Console.ReadLine();
            string set1 = Console.ReadLine();
            int a = UserProgramCode.FindPermutations(set,set1);
            Console.WriteLine(a);

        }
    }
}
```

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static int FindPermutations(string s,string n)
        {
            int c=0;
            string[] a=s.Split(' ');
            string[] b=n.Split(' ');
            foreach (string q in a)
            {

                if (b[1] == q)
                {
                    c++;
                }
                else
                    return 0;

            }
            return c;

        }
    }
}
```

Finding common Elements in multiples of 3

Write a program to find the common elements from all the three input integer lists which are also multiple of 3. Sort the result in descending order and print it.

Include a class **UserProgramCode** with static method FindCommonElements() which accepts 3 integer List.

The return type is List<int> which returns common elements in the List.

Example :
Input :
3 ---------->First list size
21
6
8
2---------->Second list size
21
6
4---------->Third list size
4
89
21
6
Output :
21
6

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

Input and Output Format:
Refer Example for input Format.
Output is a list which contains the common elements in multiples of 3 or a string as specified below.

In **FindCommonElements()**

- If there is no common elements found in all of three input lists then assign 0 to the first element of the output list and return the list.
- If any of the input lists have negative element then then assign -1 to the first element of the output list and return the list.
- If any of the input lists have element value greater than 500 then assign -2 to the first element of the output list and return the list.

In **Program** class
If the method returns a list with the first element being 0, then print "No match found".
If the method returns a list with the first element being -1, then print "The list contains negative values".
If the method returns a list with the first element being -2, then print "The elements of the list should be less than or equal to 500".
Otherwise print the result.

SAMPLE INPUT 1:

3
21
6
8
2
21
9
4
4
89
21
56

SAMPLE OUTPUT 1:

21

SAMPLE INPUT 2:

4
13
-27
44
9
5
24
9
41
56
8
4
27
24
-9
8

SAMPLE OUTPUT 2:

The list contains negative values

SAMPLE INPUT 3:

3
33
27
444
5
24
9
41
56
8
4
27
24
78
55

SAMPLE OUTPUT 3:

No match found

SAMPLE INPUT 4:

8
3
666
7
4
9
24
21
8
7
16
17
24
9
21
56
8
6
3
6
7
24

9
8

SAMPLE OUTPUT 4:

The elements of the list should be less than or equal to 500

Matching String

Write a program to display the strings which starts with the character passed in input2 variable from input1 list irrespective of case. Store the result in output list in the same sequence in which they are found in input1 list. Then form a string in output1 list as in the example given below. In the example given below, 2 corresponds to the numer of strings in the input list that start with the given character.

Example:

input1: [abc,apple,Mango]
input2: a

Output1:[abc_2,apple_2]

Business Rule:

1. If there is no match found in input1 list then store "-1" in the string list returned from from the method and print “No match found” in Main.

Include a class UserProgramCode with a static method sortStrings which accepts an integer (the size of the string list), a String list and a character. The return type (String List) should return the output String List.

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

Input and Output Format:

Input consists of an integer, a String list and a character, where the integer corresponds to the size of the List, String list corresponds to the input string list and the character values corresponds to the starting character.
Output consists of a String List.

Refer sample output for formatting specifications.

Sample Input 1:

3
abc
apple

Mango
a
Sample Output 1:
abc_2
apple_2

Sample Input 2:
3
abc
apple
Mango
b
Sample Output 2:
No match found

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            string[] a = new string[n];
            for (int i = 0; i < n; i++)
            {
                a[i] = Console.ReadLine();
            }
            char c=char.Parse(Console.ReadLine());

            List<string> y = UserProgramCode.FindPermutations(a,c);
            foreach (string j in y)
            {
                Console.WriteLine(j);
            }

        }
    }
}

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

namespace ConsoleApplication1
```

```
{
    class UserProgramCode
    {
        public static List<string> FindPermutations(string[] s,char n)
        {
            StringBuilder sb=new StringBuilder();
            List<string> q = new List<string>();
            List<string> h = new List<string>();
            string res="";
            int c=0;
            foreach (string w in s)
            {
                char[] ch = w.ToCharArray();
                if (ch[0] == n)
                {
                    q.Add(w);
                    c++;
                }
                else
                    q.Add("-1");
                return q;

            }
            res=c.ToString();
           foreach(string b in q)
           {
               sb.Append(b);
               sb.Append("_");
               sb.Append(c);
               sb.Append(' ');
           }
           res = sb.ToString();
           string[] f = res.Split(' ');
           foreach (string mn in f)
           {
               h.Add(mn);
           }
           return h;

        }
    }
}
```

Sum of Squares of Digits

Write a program to find out the sum of squares of digits in a given number.

Example:

input = 321

output = (3*3+2*2+1*1) = 14

Include a class **UserProgramCode** with a static method **getSumOfSquaresOfDigits** that accepts an integer and returns an integer.

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

Input and Output Format:
Input consists of an Integer.
Output consists of an integer.

Sample Input 1:
123

Sample Output 1:
14

Sample Input 2:
102

Sample Output 2:
5

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

namespace ConsoleApplication7
{
   class Program
   {
      static void Main(string[] args)
      {
         int n = int.Parse(Console.ReadLine());

         int e = usermainprogram.sumofsquares(n);
         Console.WriteLine(e);
      }
   }
}
```

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

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

namespace ConsoleApplication7
{
   class usermainprogram
   {
      public static int sumofsquares(int n)
      {
         int d = 0;
         int sum = 0;
         while (n > 0)
         {
            d = n % 10;
            sum = sum + (d * d);
            n = n / 10;
         }
         return sum;
      }
   }
}
```

Calculate Simple Interest

Write a customized Simple Interest Calculator to calculate simple interest for any given inputs of Principal P and years N for a private bank in Europe.

Business Rules:

1) If principal is greater than or equal to 99999 & less than 500001 and years > 5, the interest rate is 8.75%.
2) If principal is greater than or equal to 500000 & less than 1000001 and years > 3, the interest rate is 9.25%.
3) Any other principal amount and years will attract the standard interest rate of 8.25%.
4) The bank can accept max of 1000000 for a deposit. For amount greater than 1000000 the calculator should return -1.
5) Round off the interest to 2 decimal places.
6) Simple interest formula: (P*N*R) /100.
7) All the inputs should be non negative values. Else return -1.

Include a class UserProgramCode with a static method calculateSimpleInterest which accepts two integers and returns a double.
The first input parameter refers to the principal P and the second input parameter refers to the years N.

The return type (Double) should return Simple Interest amount. Refer Business Rules and return -1accordingly.

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

Input and Output Format:
Input consists of two integers, the principal amount and the number of years, respectively.
Output consists of an Integer (the gift voucher amount) or a String “Interest cannot be calculated” if any of the inputs is invalid.

Refer sample output for formatting specifications.

Sample Input 1:
100000
1
Sample Output 1:
8250

Sample Input 2:
10000000
1
Sample Output 2:
Interest cannot be calculated

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

namespace ConsoleApplication7
{
   class Program
   {
      static void Main(string[] args)
      {
         int p = int.Parse(Console.ReadLine());
         int y = int.Parse(Console.ReadLine());

         double e = usermainprogram.sumofsquares(p,y);

         if (e==-1)
```

```
            {
                Console.WriteLine("Interest cannot be calculated");
            }
            else
                Console.WriteLine(e);

        }
    }
}


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

namespace ConsoleApplication7
{
    class usermainprogram
    {
        public static double sumofsquares(int p ,int y)
        {
            double r;
            double s;
            double j;
            if (p < 0 || p>1000000 || y < 0)
            {
                return -1;
            }
            if (p >= 99999 && p < 500001 && y > 5)
               r = 8.75;
            if (p >= 500000 && p < 1000001 && y > 3)
                r = 9.25;
            else
                r = 8.25;


            s = (p * y * r) / 100;
            j=Math.Round(s,2);
            return j;
```

```
    }
  }
}
```

Next Year Day

Write a program to read a date String in dd/mm/yyyy format and to calculate the day which falls on the same date next year and print it. Note - return the output in small case.

Example:
Input = 13/07/2012
output = saturday

Include a class UserProgramCode with a static method nextYearDay which accepts a String. The return type (String) should return the day which falls on the same date next year. Return -1 in case the format of the date is incorrect.

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 a String, date in dd/mm/yyyy format.
Output consists of a String, the the day which falls on the same date next year.

Refer sample output for formatting specifications.

Sample Input:
13/07/2012
Sample Output:
Saturday

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string q = Console.ReadLine();
            string w = UserProgramCode.FindPermutations(q);
            Console.WriteLine(w);

        }
```

```
    }
}

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static string FindPermutations(string s)
        {
            string format = "dd/MM/yyyy";
            DateTime dt,dt1;
           bool r= DateTime.TryParseExact(s, format, null,
System.Globalization.DateTimeStyles.None, out dt);
           if (r == false)
           {
               return "-1";
           }
            dt1=dt.AddYears(1);
            return dt1.DayOfWeek.ToString();

        }
    }
}
```

Validating the pan

Write a code to validate the given PAN number as per the following rules:
1. There must be eight characters.
2. First three letters must be alphabets followed by four digit number and ends with alphabet
3. All alphabets should be in uppercase.

Include a class **UserProgramCode** with static method **validatePAN** that accepts the String and return type should be interger.
Create a class **Program** which would get the input and call the static method **validatePAN** present in the**UserProgramCode**.

In **validatePAN()** if PAN card number is Valid return 1 otherwise return 2 .
In **Program**

- If the method returns 1 then print "Valid PAN code"
- If the method returns 2 then print "Invalid PAN code"

Sample Input 1
ALD3245E

Sample Output 1
Valid PAN code

Sample Input 2
4567123
Sample Output 2
Invalid PAN code

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string q = Console.ReadLine();
            string w = UserProgramCode.FindPermutations(q);
            Console.WriteLine(w);

        }
    }
}
```

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static string FindPermutations(string s)
        {
            Regex reg = new Regex(@"^([A-Z]{3}[0-9]{4}[A-Z]{1})$");
            if (reg.IsMatch(s))
            {
                return "1";
            }
            else
                return "2";

        }
    }
```

```
}
```

Convert Format

Write a program to convert a 10 digit positive number which is in the format XXX-XXX-XXXX to the format XX-XX-XXX-XXX.

Include a class **UserProgramCode** with a static method **convertFormat** that accepts a string and returns a string.

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

Input and Output Format:

Input consists of a String.

Output consists of a string

Sample Input 1:

555-555-0000

Sample Output 1:

55-55-550-000

Sample Input 2:

000-000-0000

Sample Output 2:

00-00-000-000

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

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

```
            string q = Console.ReadLine();
            string w = UserProgramCode.FindPermutations(q);
            Console.WriteLine(w);

        }
    }
}

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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static string FindPermutations(string s)
        {
            StringBuilder sb=new StringBuilder();
            StringBuilder sb1 = new StringBuilder();
            char[] ch = s.ToCharArray();
            for(int i=0;i<ch.Length;i++)
                if (char.IsDigit(ch[i]))
                {
                    sb.Append(ch[i]);
                }
            string p = sb.ToString();
            sb1.Append(p.Substring(0, 2));
            sb1.Append("-");
            sb1.Append(p.Substring(2, 2));
            sb1.Append("-");
            sb1.Append(p.Substring(4, 3));
            sb1.Append("-");
            sb1.Append(p.Substring(7, 3));

            return sb1.ToString();



        }
    }
}
```

Calculate Bill Amount

A Electrical shop has announced the following seasonal discount for the purchase of certain items.

IInclude a class **UserProgramCode** with a static method **calculateBillAmount** which accepts double and char value as input and returns an integer that corresponds to the net amount.

Compute the net amount to be paid by the customer based on the below criteria,

Purchase Amount(Rs)	Discount on Tv	Discount on MusicSystem
1-25000	5%	10%
25001-50000	10%	20%
More than 50000	15%	30%

[Hint: DiscountPrice=(DiscountRate/100)*Amount of Purchase.
Net Amount=Amount of Purchase-DiscountPrice.]

If the purchase item is other than TV and MusicSystem return -2,
if the purchase amount is a negative value return -1.Otherwise return the net amount.

Create a **Main** class which gets double and char as an input and call the static method **calculateBillAmount** present in the**UserProgramCode.**
If the method returns -1, then print 'Negative Values'.
If the method returns -2, then print 'No Items'.

Input and output format:
The input will be a double and char values.
If the input character is 'T', it corresponds to TV.
If the input character is 'M', it corresponds to Music System.

Refer sample output for formatting specifications.

Sample Input 1:
20000
X
Sample Output 2:
No Items

Sample Input 2:
-5000
M
Sample Output 2:
Negative Values

Sample Input 3:
70000

T

Sample Output 3:

59500

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

namespace ConsoleApplication7
{
   class Program
   {
      static void Main(string[] args)
      {
         double a = double.Parse(Console.ReadLine());
         char p = char.Parse(Console.ReadLine());

         int e = usermainprogram.calculatebillamount(a,p);

         if (e==-1)
         {
            Console.WriteLine("negative values");
         }
         else
         if (e == -2)
         {
            Console.WriteLine("no items");
         }
         else
            Console.WriteLine(e);

      }
   }
}
```

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

```
namespace ConsoleApplication7
{
    class usermainprogram
    {
        public static int calculatebillamount(double a  ,char p)
        {
            double discount=0;
            double gross;
            int total;
            if (a < 0)
            {
                return -1;
            }
            if (p != 'T' && p != 'M')
            {
                return -2;
            }

            if (a > 0 && a < 25001 && p == 'T')
            {

                discount = 0.05 * a;
            }
            else
                if (a > 0 && a < 25001 && p == 'M')
                {
                    discount = 0.1 * a;
                }
                else

            if (a > 25000 && a < 50001 && p == 'T')
            {
                discount = 0.1 * a;
            }
            else
                if (a > 25000 && a < 50001 && p == 'M')
                {
                    discount = 0.2 * a;
                }
                else

                    if (a > 50000 && p == 'T')
                    {
```

```
                discount = 0.15 * a;
            }
            else

                if (a > 50000 && p == 'M')
                {
                    discount = 0.3 * a;
                }
        gross = a - discount;

        total = Convert.ToInt32(gross);

        return total;




    }
  }
}
```

Close

Sum Non Prime Numbers

Write a program to calculate the sum of all the non prime positive numbers less than or equal to the given number.

Note: prime is a natural number greater than 1 that has no positive divisors other than 1 and itself

Example:
input = 9
Prime numbers = 2,3,5 and 7
output = 1+4+6+8+9=28

Include a class **UserProgramCode** with a static method "**addNumbers**" that accepts an integer arguement and returns an integer.

Create a class **Program** which would get an integer as input and call the static method **addNumbers** present in the **UserProgramCode**.

Input and Output Format:
Input consists of an integer.

Output consists of an integer.

Sample Input:

9

Sample Output:

28

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

namespace ConsoleApplication19
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = int.Parse(Console.ReadLine());
            int y = userprogramcode.prime(n);
            Console.WriteLine(y);

        }
    }
}

namespace ConsoleApplication19
{
    class userprogramcode
    {

        public static int prime(int n)
        {
            int count = 0, sum = 1;

            int i, j;
            for (i = 2; i <= n; i++)
            {
                count = 0;
                for (j = 1; j <= i; j++)
                {
                    if (i % j == 0)
                    {
                        count++;
                    }
                }
                if (count != 2)
                {
                    sum = sum + i;
                }
            }
            return sum;
}
    }
```

}

Odd Even Sum

Write a program to compare the sum of digits at even indexes (say evenSum) and sum of digits at odd indexes (say oddSum) in the given number.

Example1:

Input1: 23050
(evenSum = 2 + 0 + 0 = 2
oddSum = 3 + 5 = 8)

Output = -1

Example2:

Input1: 23111
(evenSum = 2 + 1 + 1 = 4
oddSum = 3 + 1 = 4)

Output = 1

Business Rule:
1. If both the sums are same then print 1 else print -1.

Create a class named UserProgramCode that has the following static method
public static int sumOfOddEvenPositioned(int input1)

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

Input and Output Format:

Input consists of an integer.

Output is either 1 or -1.

Sample Input :

23050

Sample Output :

-1

Extract Max Substring

Write method to get a the substring with maximum number of characters for given string input (input1) separated by given delimeter (input2). If two or more substrings have maximum number of characters return the substring which appears first in the alphabetical order.

Example :
Input1 = "delhi-pune-patna"
Input2 = "-"
Output1 = "delhi"

Create a class named UserProgramCode that has the following static method
public static string extractMax(string input1, string input2)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
Input consists of 2 strings.
Output is a string.

Sample Input :
delhi-pune-patna
-

Sample Output :
delhi

Arrange After Cubing

Write a code to insert cube of a number in between two numbers in an integer array if those numbers satisfy the below conditions:

Conditions:
1) The elements in the array must be consecutive numbers
2) Second element in the array should be the square of first element in the array.
3) Cube of first number must be inserted in between the first element and second element.

4) If any of the element in the array does not satisy the above condition add the element in the output array and proceed with the next element.

Business Rules:
1. If no consecutive numbers are present assign all the elements in the Input array to the Output array .
2. If input array consists of negative elements perform the square of negative numbers and then print the final output array.

Create a class named UserProgramCode that has the following static method
public static int[] arrangeAfterCubing(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:

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

The next 'n' lines of input consist of elements in the input array.

Refer sample output for formatting specifications.

Sample Input 1:

7
1
2
4
6
7
3
9

Sample Output 1:

1
2
8
4
6
7
3
27
9

Sample Input 2:

3
1

-4
5

Sample Output 2:
1
16
5

Count Even Occurrence

Given an int array, write a program to calculate the count as follows.
If the same element is repeated even number of times,increase the count by one.
Print the value of the count.

Business Rules :
1. If any of the element in the array is a negative number, then print -1.
2. If there is no elements repeated in even number of times, then print 0.

Create a class named UserProgramCode that has the following static method
public static int countEvenOccurence(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.
The next 'n' lines of input consist of elements in the input array.
Output consists of an integer.
Refer sample output for formatting specifications.

Sample Input 1 :
17
1
2
3
4
9
3
6
7
1

9
100
2
4
1
45
1
9

Sample Output 1 :
4

Sample Input 2 :
13
1
2
3
4
3
6
7
1
9
100
2
4
-17

Sample Output 2 :
-1

Repeated Words

Given two string inputs input1 and input2, write a program to calculate the number of times each word in input1 occurs in input2 and print the words which has occurred the maximum number of times in the output with empty spaces between them and also in the same order as given in input1.
Output should be printed in lower case.Ignore case sensitiveness in both the input strings.

Business Rules:
1)If any of the words is repetitive in input1, then print -1.

2)If none of the words from input1 has occurred in input2, then print -2.

Create a class named UserProgramCode that has the following static method
public static string repeatedWords(string input1, string input2)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
Input consists of 2 strings, input1 and input 2.
Output consists of a string.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
Apple is a fruit
apple is a nice fruit and apple is available in all seasons

Sample Output 1 :
apple is

Sample Input 2 :
Does he have to have a car ?
Yes he should.

Sample Output 2 :
-1

Max Diff in Array

Given an integer input array input,find out the maximum difference between any two elements such that larger element appears after the smaller number in the input array.

Business Rules :
1. If any of the given inputs contain any negative number, then print -1.
2. If there is only one element or more than 10 elements in the input array, then print -2.
3. If there are any duplicates in the input array, then print -3.

Create a class named UserProgramCode that has the following static method
public static int diffIntArray(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:

The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.
The next 'n' lines of input consist of elements in the input array.
Output is an integer.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
7
2
3
10
6
4
8
1

Sample Output 1 :
8

[Hint : (Diff between 2 and 10. 10 is larger than 2)]

Sample Input 2 :
5
4
5
-20
9
10

Sample Output 2 :
-1

Bonus Calculation

Write a program to calculate the bonus of the employee given the basic salary of the employee.

The bonus will be calculated based on the below category.

If Basic Salary>15000 and less than 20001 calculate bonus as 17% of basic+1500
If Basic Salary>10000 and less than 15001 calculate bonus as 15% of basic+1200
If Basic Salary<10001 calculate bonus as 12% of basic+1000

for rest calculate bonus as 8%of basic+500

Business rule:
1) If the salary given is a negative number, then print -1.
2) If the salary given is more than 1000000, then print -2 .
3) All the test cases has the calculated bonus as integer value only.

Create a class named UserProgramCode that has the following static method
public static int calculateBonus(int input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
Input consists of an integer that corresponds to the salary.
Output is an integer.
Refer sample output and business rule for output formatting specifications.

Sample Input 1 :
10000

Sample Output1 :
2200

Sample Input 2 :
2000000

Sample Output 2 :
-2

Close

GCD – Array

Given an array of integers as input, write a program to find the Greatest Common Divisor for all the integer elements present in the input.

Greatest Common Divisor also known as the greatest common factor (gcf), of two or more integers is the largest positive integer that divides the numbers without a remainder.

Business Rule:
1. If the input array contains any value less than 1, assign -1 to the output1 variable.

Create a class named UserProgramCode that has the following static method
public static int greatestCommonDivisor(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:

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

The next 'n' lines of input consist of elements in the input array.

Output is an integer.

Refer business rules and sample output for formatting specifications.

Sample Input 1 :

4
24
12
20
8

Sample Output 1 :

4

Sample Input 2 :

4
2
4
8
-6

Sample Output 2 :

-1

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

namespace ConsoleApplication1
{
    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 w = UserProgramCode.FindPermutations(a);
            Console.WriteLine(w);

        }
    }
}


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

namespace ConsoleApplication1
{
    class UserProgramCode
    {
        public static int FindPermutations(int[] a)
        {


            int l=a.Length;
            int count=0;
            int m = a.Min();
            while (m>0)
            {
                count = 0;
                for (int i = 0; i < l; i++)
                {
                    if (a[i] % m == 0)
                    {
                        count++;
                    }



                }
                if (count == l)
                {
                    return m;

                }
                else
                    m--;

            }
            return m;
        }
```

```
    }
}
```

Close

Train Tariff Calculation

Ram has to book his train tickets for travelling from Chennai to Pune through an online portal.
The tariffs vary based on parameters such as the Date of booking(DOB) and the Date of Travel(DOT).
Consider the number of days between DOB and DOT as NOD.

The normal ticket cost from Chennai to Pune when the ticket booking is done minimum one month before is Rs 1000 in Sleeper class(SL). The normal ticket cost for AC class ratings are as follows :

First Class AC (1AC):Rs 2500
Second Class AC(2AC): Rs 2000
Third Class AC(3AC) : Rs 1500

A. If NOD is from 21 days upto 30 days ,then the tariff is 10% more than normal ticket cost
B. If NOD is from 11 days upto 20 days ,then the tariff is 20% more than normal ticket cost
C. If NOD is from 4 days upto 10 days ,then the tariff is 30% more than normal ticket cost
D.If NOD is upto 3 days ,then the tariff is 40% more than normal ticket cost.

Write a program to calculate the total cost a person has to pay for their booking given their date of booking , the date of travel and the class of travel as input1, input2 and input3 respectively and print the output in the following format.

Your ticket is confirmed and the booking cost is Rs YYYYYY

where YYYYYY is the calculated booking cost. Print the booking cost as an integer.

The date of booking and the date of travel are given as string in the format yyyy.mm.dd

Business rules:
1. If the date of travel is less than 3 days from the date of booking, then the tickets cannot be booked and print “Short Notice and hence Tickets cannot be booked” .
2. If the date of booking or the date of travel are not in the date format, then print “Improper Date format in the input” .
3. If the date of travel is more than 90 days from the date of booking, then the tickets cannot be booked and print “Long Notice period and hence Tickets cannot be booked” .
4. If the class of travel is other than SL,1AC,2AC or 3AC, then print “Improper class of Travel” .

Create a class named UserProgramCode that has the following static method
public static int calculateTrainTariff(string input1, string input2, string 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 strings --- input1 (Date of Booking), input2 (Date of Travel) and input3(class of travel).
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
2014.05.15
2014.05.25
SL

Sample Output 1 :
1200

Sample Input 2 :
201405.15
2014.05.25
SL

Sample Output 2 :
Improper Date format in the input

Close

Array Median

Given an integer array as input, write a program to calculate median of the given numbers in the array with the below conditions
1. Sort the Input array in ascending.
2. When the number of elements in the Input array is odd, then median will be the middle number. Median is the (N+1)/2th element.
3. When the number of elements in the Input array is even, then median will be the average of two middle numbers. Round off the median value to the nearest integer.

Business rule:
1) If the Input array contains any negative numbers, then print -1.
2) If any of the Input array elements contains "0", then print -2.

Example1:
Input :
7
1
2
1
4
7
1
2
Output :
2
After sorting the array is {1,1,1,2,2,4,7} Number of element in input array N is 7
(N+1)/2th element= (7+1)/2= 4th element
Median value is the 4th number which is 2

Example 2:
Input :
6
52
51
81
84
60
88
Output :
71
After sorting the array is {51,52,60, 81, 84, 88}
Median = Average of Middle 2 numbers = (60 + 81)/2 = 70.5.

round(70.5) = 71

Create a class named UserProgramCode that has the following static method
public static int calculateMedian(int[] input1)
Create a class named Program that accepts the inputs and calls the static method present in the UserProgramCode.

Input and Output Format:
The first line of the input consists of an integer, n that corresponds to the number of elements in the input array.
The next 'n' lines of input consist of elements in the input array.
Refer business rules and sample output for formatting specifications.

Sample Input :
7
1
2
1
4
7
1
2

Sample Output :
2

Close

Student Score

Given a string array Input (Input 1) containing Student name and percentage of marks in the below format Input = {StudentName1, Mark1, StudentName2, Mark2, StudentName3, Mark3,.....etc},
write a program to determine the student grade based on below condition, and print the output in the below format.

AAA has scored BBB marks with CCC scores

where AAA - Input StudentName (Input 2), BBB - Mark and CCC - Grade in Upper case.

Grade calculation:
If the mark is greater than or equal to 80, then OUTSTANDING

If the mark is less than 80 and greater than or equal to 60, then GOOD
If the mark is less than 60 and greater than or equal to 50, then AVERAGE
If the mark is less than 50, then FAIL

Business rule:
1) If any of the StudentName in Input1 or Input2 contains any special characters, then print “Invalid Input”.
2) If the Input2 string value is not present in Input1 array, then print “Invalid Student”
3) If the Input1 array length is odd, then print “No corresponding Student or Mark”

Create a class named UserProgramCode that has the following static method
public static string studentScore(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 input array.
The next 'n' lines of input consist of elements in the input array.
The next line of the input consists of a string that corresponds to the student name.
Refer business rules and sample output for formatting specifications.

Sample Input 1 :
4
Ram
55
Vignesh
89
Vignesh

Sample Output 1 :
Vignesh has scored 89 marks with OUTSTANDING grade

Sample Input 2 :
5
Anil
76
Sunil
68
Raja
Vignesh

Sample Output 2 :
No corresponding Student or Mark

Close

Berth Type

Ram books 3 train tickets for his father,grandfather and himself.Their seat numbers are given as input1,input2 and input3 respectively.
Each person is assigned a seat based on the berth type preference(Lower [L], Middle [M], Upper [U], SideLower [SL] and SideUpper[SU]).

The berth type can be identified based on the following steps:
1. Divide the seat number by 8
2. If the remainder is either 1 or 4, then the berth type is Lower
If the remainder is either 2 or 5, then the berth type is Middle
If the remainder is either 3 or 6, then the berth type is Upper
If the remainder is 7, then the berth type is SideLower
If the remainder is 0, then the berth type is SideUpper
3. Based on the grandfather's berth type, print the following:
If berth type is Lower(L), print
Lower berth provided as per request
If berth type is other than Lower(L),swap it with other seat numbers which is lower and print
Your seat has been swapped from XX to YY as per preference request
If berth type is not Lower(L) for all the three seats,print
Your seat will be changed on the date of travel

here XX is the initial seat number of Ram's grandfather and YY is the swapped seat number.First try swapping with Ram's seat if found lower,else with Ram's fathers seat.Maximum seat number limit - 1000.

Business rules:
1.If any of the seat number is zero or negative , print “Invalid Seat Number”.
2.If any of the seat numbers contain any alphabets or special characters , print “Invalid Input”

Create a class named UserProgramCode that has the following static method
public static string checkBerthType(string input1,string input2,string 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 strings – input1 (Ram's father's berth number), input2 (Ram's grandfather's berth number) and input3 (Ram's berth number).

Output consists of a string.

Refer business rules and sample output for formatting specifications.

Sample Input 1 :

76
75
78

Sample Output 1 :

Your seat has been swapped from 75 to 76 as per preference request

Sample Input 2 :

76
-75
78

Sample Output 2 :

Invalid Seat Number