61.String Processing - ZigZag

Write a program to read a string containing date in DD-MM-YYYY format. find the number of days in the given month.

Note - In leap year February has got 29 days.

Include a class UserMainCode with a static method **getLastDayOfMonth** which accepts the string. The return type is the integer having number of days.

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

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

Sample Input 1:
12-06-2012
Sample Output 1:
30

Sample Input 2:
10-02-2012
Sample Output 2:
29

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.*;
```

```
public class User {



        public static void main(String[] args) throws IOException, ParseException  {

                // TODO Auto-generated method stub

        String s1="10-02-2012";

        SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");

        Calendar cal=Calendar.getInstance();

        Date d1=sdf.parse(s1);

        cal.setTime(d1);

        int n=cal.getActualMaximum(Calendar.DAY_OF_MONTH);

        System.out.println(n);

        }

}
```

62. Leap Year

Write a program to read a string containing date in DD/MM/YYYY format and check if its a leap year. If so, return true else return false.

Include a class UserMainCode with a static method **isLeapYear** which accepts the string. The return type is the boolean indicating TRUE / FALSE.

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

Input and Output Format:

Input consists of a string.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

Sample Input 1:

23/02/2012

Sample Output 1:

TRUE

Sample Input 2:

12/12/2011

Sample Output 2:

FALSE

```
            import java.text.ParseException;

import java.util.*;
publicclass Main
{
      publicstaticvoid main(String[] args) throws ParseException  {

                        Scanner sc=new Scanner(System.in);
                        String s=sc.nextLine();
                        System.out.println(User.leapYear(s));
                        }
            }
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

publicclass User
{
      publicstaticboolean leapYear(String s) throws ParseException
      {
            SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
      GregorianCalendar g=new GregorianCalendar();

      Calendar cal=Calendar.getInstance();
      Date d1=sdf.parse(s);
      cal.setTime(d1);
      intn=cal.get(Calendar.YEAR);

      boolean b=g.isLeapYear(n);
return b;


}}
```

63. Largest Chunk

Write a program to read a string and return the length of the largest "chunk" in the string.
A chunk is a repetition of same character 2 or more number of times. If the given string doest not contain any repeated chunk of characters return -1.
Include a class UserMainCode with a static method **getLargestSpan** which accepts the string. The return type is the integer.
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

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

Sample Input 1:

This place is soooo good

Sample Output 1:

4

```
import java.util.*;

public class Main {

public static void main(String[] args) {

String s1="You are toooo good";

System.out.println(maxChunk(s1));

}

public static int maxChunk(String s1) {

int max=0;

StringTokenizer t=new StringTokenizer(s1," ");

while(t.hasMoreTokens()){

String s2=t.nextToken();

int n=0;

for(int i=0;i<s2.length()-1;i++)

if(s2.charAt(i)==s2.charAt(i+1))

n++;

if(n>max)

max=n;

}

return (max+1);

}

}
```

64. Largest Span

Write a program to read a integer array, find the largest span in the array.
Span is the count of all the elements between two repeating elements including the repeated elements.
Include a class UserMainCode with a static method **getLargestSpan** which accepts the integer array. The return type is integer.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

Input and Output Format:
Input consists of an integer n which is the number of elements followed by n integer values.
Output consists of integer.
Refer sample output for formatting specifications.

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

```
public class Main {

public static void main(String[] args) {

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

System.out.println(maxSpan(a));

}
public static int maxSpan(int[] a) {

String s2 = null;

int n=0;

StringBuffer sb=new StringBuffer();

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

sb.append(String.valueOf(a[i]));

String s1=sb.toString();

for(int i=0;i<s1.length();i++)

for(int j=i+1;j<s1.length();j++)

if(s1.charAt(i)==s1.charAt(j))

s2=String.valueOf(s1.charAt(j));

int n1=s1.indexOf(s2);

int n2=s1.lastIndexOf(s2);

for(int i=n1+1;i<n2;i++)

n++;

return (n+2);

}

}
```

65.Even Sum & Duplicate Elements

Write a program to read a integer array, Remove the duplicate elements and display sum of even numbers in the output. If input array contain only odd number then return -1.
Include a class UserMainCode with a static method **sumElements** which accepts the integer array. The return type is integer.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

Input and Output Format:
Input consists of an integer n which is the number of elements followed by n integer values.
Output consists of integer.
Refer sample output for formatting specifications.

Sample Input 1:
7
2
3
54
1
6
7
7
Sample Output 1:
62

Sample Input 2:
6
3
7
9
13
17
21
Sample Output 2:
-1

```
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.Scanner;
public class Main
{
public static void main(String args[])
```

```
{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int a[]=new int[n];

for(int i=0;i<n;i++)
{
        a[i]=sc.nextInt();
}
System.out.println(UserMainCode.display(a));
}}


import java.util.Iterator;
import java.util.LinkedHashSet;

public class UserMainCode {
        public static int display(int a[])
        {
                LinkedHashSet<Integer>h1=new LinkedHashSet<Integer>();
                int s=0;
                for(int i=0;i<a.length;i++)
                {
                        h1.add(a[i]);
                }
                Iterator<Integer> it=h1.iterator();
                while(it.hasNext())
                {
                        int k=it.next();
                        if(k%2==0)
                        {
                                s=s+k;
                        }
                }
                if(s>0)
                        return s;
                else
                        return -1;
                }}
```

66.Regular Expression - III

Given a string (s) apply the following rules.
I)At least 8 characters must be present
II)At least one capital letter must be present
III)At least one small letter must be present
Iv)At least one special symbol must be present
V)At least one numeric value must be present
If the condition is satisifed then print valid else print invalid.

Include a class UserMainCode with a static method **passwordValidation** which accepts the string. The return type is the string.
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format:
Input consists of a string.
Output consists of string (valid / invalid) .
Refer sample output for formatting specifications.

Sample Input 1:
Technology$1213
Sample Output 1:
valid

```
public class UserMainCode

{

    public static int display(String s)

    {
            if(s.matches(".*[0-9]{1,}.*") && s.matches(".*[@#$]{1,}.*") && s.length()>=8 &&
s.matches(".*[A-Z]{1,}.*") && s.matches(".*[a-z]{1,}.*"))

                return 1;

            else

                return -1;

    }}
```

```
import java.util.*;
publicclass Main
{
      publicstaticvoid main(String[] args)      {

                        Scanner sc=new Scanner(System.in);
                        String s=sc.nextLine();
                        System.out.println(User.leapYear(s));

                        }
            }

publicclass User{
publicstaticint leapYear(String s)
      {

if(s.matches(".*[0-9]{1,}.*")
            &&s.matches(".*[!@#$%^&*]{1,}.*") &&s.length()>=8 &&
            s.matches(".*[A-Z]{1,}.*") &&s.matches(".*[a-z]{1,}.*"))
return 1;
else
return -1;
      }}
```

67.Integer Factorial

Give an array of integer as input, store the numbers and their factorials in an hashmap and print the same.
Include a class UserMainCode with a static method **getFactorial** which accepts the integer array.

The return type is the hashmap which is printed key:value.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

Input and Output Format:
Input consists of a number denoting the size of the array and followed by the elements.
Output consists of a hashmap printed in the output format .
Refer sample output for formatting specifications.

Sample Input1:
```
4
2
3
5
4
```
Sample Output1:
```
2:2
3:6
5:120
4:24
```

```
import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;



import java.util.Scanner;



public class kapes3 {

public static void main(String []args){

Scanner sc=new Scanner(System.in);

int s=Integer.parseInt(sc.nextLine());
int []a=new int[s];

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

{

a[i]=sc.nextInt();

}

LinkedHashMap<Integer,Integer>hm2=new LinkedHashMap<Integer,Integer>();

hm2=kapes4.display(a);

Iterator<Integer> it=hm2.keySet().iterator();

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

{

int n=it.next();

int fac=hm2.get(n);

System.out.println(n+":"+fac);

}

}

}
```

```
import java.text.DecimalFormat;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;

public class kapes4

{public static LinkedHashMap<Integer,Integer> display(int[] a)

{
```

```
LinkedHashMap<Integer,Integer>hm=new LinkedHashMap<Integer,Integer>();

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

{

int u=1;

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

{

u=u*j;

}

hm.put(a[i],u);

}

return hm;

}}
```

68. String processing – Long + Short + Long

Obtain two strings S1,S2 from user as input. Your program should form a string of "long+short+long", with the shorter string inside of the longer String.
Include a class UserMainCode with a static method **getCombo** which accepts two string variables. The return type is the string.
Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

Input and Output Format:
Input consists of two strings with maximum size of 100 characters.
Output consists of an string.

Refer sample output for formatting specifications.

Sample Input 1:
Hello
Hi
Sample Output 1:
HelloHiHello

```
import java.util.StringTokenizer;


public class User {

public static void main(String[] args){

        String s1="Hi";

        String s2="Hello";

        System.out.println(capsStart(s1,s2));

}

public static String capsStart(String s1,String s2){

        StringBuffer s5=new StringBuffer();

        int q=s1.length();

        int w=s2.length();

        if(q>w)

        {

                s5.append(s1).append(s2).append(s1);

        }

        else

        {

                s5.append(s2).append(s1).append(s2);

        }
        return s5.toString();

}

}
```

69. Age for Voting

Given a date of birth (dd/MM/yyyy) of a person in string, compute his age as of 01/01/2015.

If his age is greater than 18, then println eligible else println not-eligible.

Include a class UserMainCode with a static method getAge which accepts the string value. The return type is the string.

Create a Class Main which would be used to accept the two string values and call the static method present in UserMainCode.

Input and Output Format:
Input consists of two string.
Output consists of a string.
Refer sample output for formatting specifications.

Sample Input 1:
16/11/1991

Sample Output 1:
eligible

```
import java.util.*;
publicclass Main
{
      publicstaticvoid main(String[] args)       {
            Scanner sc=new Scanner(System.in);
            String s =sc.nextLine();
            System.out.println(User.display(s));
            }}
```

```
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
publicclass User{

      publicstatic String display(String n)
      {

      int year=0;
       String now="01/01/2015";
      SimpleDateFormat sdf1=new SimpleDateFormat("dd/MM/yyyy");
      try
      {

      sdf1.setLenient(false);
      Calendar c1=Calendar.getInstance();
      Date d=sdf1.parse(n);
      c1.setTime(d);
      int y=c1.get(Calendar.YEAR);
      int m=c1.get(Calendar.MONTH);
      int day=c1.get(Calendar.DAY_OF_MONTH);


      Calendar c2=Calendar.getInstance();
      Date d1=sdf1.parse(now);
      c1.setTime(d1);
      int y1=c2.get(Calendar.YEAR);
      int m1=c2.get(Calendar.MONTH);
      int day1=c2.get(Calendar.DAY_OF_MONTH);

      year=y1-y;
      //System.out.println(year);
      if(m>m1)
      year--;
      elseif(m==m1)
      {if(day<day1)
      year--;
      }
      }
      catch(Exception e)
      {
      e.printStackTrace();
      }
      if(year>18)
      return"eligible";
      else
      return"not-eligible";
      }}
```

```
public class UserMainCode{
    public static int getMaxSpan(int a[]) {
        int i,j,k,count,max=0,p=0;
        int n=a.length;
        for(i=0;i<n;i++)
        {
            count=0;
            for(j=i+1;j<n;j++)
            {
                if(a[i]==a[j])
                {
                    p=j;
                }
            }
            for(k=i;k<=p;k++)
            {
                count++;
            }
            if(count>max)
            {
                max=count;
            }
        }
        return max;
        }


}
```