51) Finding the day of birth

Given an input as date of birth of person, write a program to calculate on which day (MONDAY,TUESDAY....) he was born store and print the day in Upper Case letters.

Include a class **UserMainCode** with a static method **calculateBornDay** which accepts a string as input.

The return type of the output is a string which should be the day in which the person was born.

Create a class **Main** which would get the input and call the static method **calculateBornDay** present in the UserMainCode.

Input and Output Format:

NOTE: date format should be(dd-MM-yyyy)
Input consists a date string.

Output is a string which the day in which the person was born.

Refer sample output for formatting specifications.

Sample Input 1:

29-07-2013
Sample Output 1:

MONDAY

Sample Input 2:

14-12-1992
Sample Output 2:

MONDAY

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class Main {
	public static void main(String[] args) throws ParseException {

		Scanner sc=new Scanner(System.in);
	String s1=sc.nextLine();
	System.out.println(UserMainCode.calculateBornDay(s1));
		}
```

```
		}

import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
public class UserMainCode {
	public static String calculateBornDay(String s1) throws ParseException
	{
		SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
		SimpleDateFormat sdf1=new SimpleDateFormat("EEEEE");
		Date d=sdf.parse(s1);
		String s=sdf1.format(d);
		return s.toUpperCase();
	}
}
```

52) Removing elements from HashMap

Given a HashMap as input, write a program to perform the following operation : If the keys are divisible by 3 then remove that key and its values and print the number of remaining keys in the hashmap.

Include a class **UserMainCode** with a static method **afterDelete** which accepts a HashMap as input.

The return type of the output is an integer which represents the count of remaining elements in the hashmap.

Create a class **Main** which would get the input and call the static method **afterDelete** present in the UserMainCode.

Input and Output Format:

First input corresponds to the size of hashmap

Input consists a HashMap

Output is an integer which is the count of remaining elements in the hashmap.

Refer sample output for formatting specifications.

Sample Input 1:

4
339
RON
1010

JONS
3366
SMITH
2020
TIM

Sample Output 1:

2

Sample Input 2:

5
1010
C2WE
6252
XY4E
1212
M2ED
7070
S2M41ITH
8585
J410N

Sample Output 2:

3

53) Experience Calculator

Write a program to read Date of Joining and current date as Strings and Experience as integer and validate whether the given experience and calculated experience are the same. Print “true” if same, else “false”.

Include a class **UserMainCode** with a static method **calculateExperience** which accepts 2 strings and an integer. The return type is boolean.

Create a Class Main which would be used to accept 2 string (dates) and an integer and call the static method present in UserMainCode.

Input and Output Format:

Input consists of 2 strings and an integer, where the 2 strings corresponds to the date of joining and current date, and the integer is the experience.

Output is either “true” or “false”.

Refer sample output for formatting specifications.

Sample Input 1:
11/01/2010
01/09/2014
4
Sample Output 1:
true

Sample Input 2:
11/06/2009
01/09/2014
4
Sample Output 2:
False

```
import java.util.Date;

import java.text.SimpleDateFormat;

public class Usermaincode

{public static boolean display(String s,String s1,int n)

{

boolean b=false;

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

try{

Date d=sdf.parse(s);

Date d1=sdf.parse(s1);

int y=d.getYear();

int y1=d1.getYear();

int m=d.getMonth();

int m1=d1.getMonth();

int day=d.getDay();
```

```
int day1=d1.getDay();

int age=y1-y;

if(m>m1)

age--;

else if(m==m1)

{if(day<day1)

age--;

}

if(age==n)

b=true;

else

b=false;

}

catch(Exception e)

{e.printStackTrace();

}

return b;

}

}
```

54) Flush Characters

Write a program to read a string from the user and remove all the alphabets and spaces from the String, and **only store special characters and digit** in the output String. Print the output string.

Include a class **UserMainCode** with a static method **getSpecialChar** which accepts a string. The return type (String) should return the character removed string.

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

Input and Output Format:

Input consists of a strings.

Output consists of an String (character removed string).

Refer sample output for formatting specifications.

Sample Input :
cogniz$#45Ant
Sample Output :
$#45

```
public class User {
      public static String repeatString (String s)
      {
            StringBuffer sb=new StringBuffer();
            for(int i=0;i<s.length();i++)
            {

                        /* char c=s.charAt(i);
                  if(!Character.isAlphabetic(c)) */


            // if(!Character.isAlphabetic(s.charAt(i))
&& (!charachter.isWhiteSpace(s.CharAt(i)))

if( (!Character.isAlphabetic(s.charAt(i)))
                        &&  (!Character.isWhitespace(s.charAt(i))) )

                  sb.append(s.charAt(i));
            }
            return sb.toString();
       }
}
```

55) String Repetition

Write a program to read a string and an integer and return a string based on the below rules.

If input2 is equal or greater than 3 then repeat the first three character of the String by given input2 times, separated by a space.

If input2 is 2 then repeat the first two character of String two times separated by a space,

If input2 is 1 then return the first character of the String.

Include a class UserMainCode with a static method **repeatString** which takes a string & integer and returns a string based on the above rules.

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

Input and Output Format:

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

Sample Input 1:

COGNIZANT

4

Sample Output 1:

COG COG COG COG

Sample Input 2:

COGNIZANT

2

Sample Output 2:

CO CO

```
public class User {
	public static String repeatString (String s,int n)
	{
		StringBuffer sb=new StringBuffer();
		if(n>=3)
		{
			for(int i=0;i<n;i++)
			{
			sb.append(s.substring(0,3)).append(" ");
			}
		}
		else if(n==2)
```

```
            {
                  for(int i=0;i<n;i++)
                  sb.append(s.substring(0,2)).append(" ");
            }
            else if(n==1)
            {
                  for(int i=0;i<n;i++)
                  sb.append(s.substring(0,1)).append(" ");
            }

            return sb.toString();
      }
}
```

56) Average of Prime Locations

Write a program to read an integer array and find the average of the numbers located on the Prime location(indexes).

Round the avarage to two decimal places.

Assume that the array starts with index 0.

Include a class UserMainCode with a static method **averageElements** which accepts a single integer array. The return type (double) should be the average.

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

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 a single Double value.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

Sample Input 1:

8

4

1

7

6

5

8

6

9

Sample Output 1:

7.5

```
public class User {
	public static float averageElements(int a[],int n)
	{
		int c=0,sum=0,k=0;
		float avg=0;
		for(int i=2;i<=n;i++)
		{
			c=0;
			for(int j=1;j<i;j++)
			{
			if(i%j==0)
				c++;

		}
			if(c==1)
			{
				k++;
				sum=sum+a[i];
			}
		}
		avg=(float)sum/k;
return avg;
	 }
}
```

57) Common Elements

Write a program to read two integer arrays and find the sum of common elements in both the arrays. If there are no common elements return -1 as output

Include a class UserMainCode with a static method **sumCommonElements** which accepts two single integer array. The return type (integer) should be the sum of common elements.

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

Assume that all the elements will be distinct.

Input and Output Format:

Input consists of 2n+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, The last n elements correspond to the elements of the second array.

Output consists of a single Integer value.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

Sample Input 1:

4

1

2

3

4

2

3

6

7

Sample Output 1:

5

```
public class User {
	public static int getMiddleElement (int a[],int b[],int n)
	{
		int sum=0;
		for(int i=0;i<n;i++)
		{
			for(int j=0;j<n;j++)
			{
			if(a[i]==b[j])
				sum=sum+a[i];
```

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

58) Middle of Array

Write a program to read an integer array and return the middle element in the array. The size of the array would always be odd.

Include a class UserMainCode with a static method **getMiddleElement** which accepts a single integer array. The return type (integer) should be the middle element in the array.

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

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 a single Integer value.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 19.

Sample Input 1:

5

1

5

23

64

9

Sample Output 1:

23

```
public class User {
	public static int getMiddleElement (int a[])
	{
		int n=a.length;
		int mid=n/2;
		return a[mid];

	 }
}
```

59) Simple String Manipulation

Write a program to read a string and return a modified string based on the following rules.

Return the String without the first 2 chars except when

1. keep the first char if it is 'j'
2. keep the second char if it is 'b'.

Include a class UserMainCode with a static method **getString** which accepts a string. The return type (string) should be the modified string based on the above rules. Consider all letters in the input to be small case.

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

Input and Output Format:

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

Output consists of a string.

Refer sample output for formatting specifications.

Sample Input 1:

hello

Sample Output 1:

llo

Sample Input 2:

java

Sample Output 2:

Jva

```
public class User {
	public static String getString(String s)
	{
		StringBuffer sb=new StringBuffer();
		char a=s.charAt(0);
		char b=s.charAt(1);
		if(a!='j'&& b!='b')
			sb.append(s.substring(2));
			else if(a=='j' && b!='b')
			sb.append("j").append(s.substring(2));
			else if(a!='j' && b=='b')
			sb.append(s.substring(1));
			else
			sb.append(s.substring(0));
			return sb.toString();

	 }
}
```

60) Date Validation

Write a program to read a string representing a date. The date can be in any of the three formats

1:dd-MM-yyyy 2: dd/MM/yyyy 3: dd.MM.yyyy

If the date is valid, print **valid** else print **invalid**.

Include a class UserMainCode with a static method **getValidDate** which accepts a string. The return type (integer) should be based on the validity of the date.

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

Input and Output Format:

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

Sample Input 1:

03.12.2013

Sample Output 1:

valid

Sample Input 2:

03$12$2013

Sample Output 3:

Invalid

```
public class User {
	public static int getValidDate(String s) throws ParseException
	{
		int res=0;
		// String s1=null,s2=null,s3=null;
		if(s.matches("[0-9]{2}[.]{1}[0-9]{2}[.]{1}[0-9]{4}"))
		{
	   SimpleDateFormat sdf1=new SimpleDateFormat("dd.MM.yyyy");
	   sdf1.setLenient(false);
	   try
	   {
	   Date d1=sdf1.parse(s);
	   res=1;
	   }
	   catch (ParseException e)
	   {
		  res=-1;
	   }
	   }

		else if(s.matches("[0-9]{2}[-]{1}[0-9]{2}[-]{1}[0-9]{4}"))
		{

                                SimpleDateFormat sdf3=new SimpleDateFormat("dd-MM-
yyyy");
                                sdf3.setLenient(false);
                                try
                                {
                                Date d1=sdf3.parse(s);
                                res=1;
                                }
                                catch (ParseException e)
                                {
                                      res=-1;
                                }
		 }
		else if(s.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{4}"))
		{

                                SimpleDateFormat sdf3=new
SimpleDateFormat("dd/MM/yyyy");
                                sdf3.setLenient(false);
                                try
                                {
                                Date d1=sdf3.parse(s);
                                res=1;
                                }
                                catch (ParseException e)
                                {
                                      res=-1;
                                }
```

```
            }

        else
            res=0;

      return res;
    }
}
```