Set-2

1.No of days

Given two inputs year and month (Month is coded as: Jan=0, Feb=1 ,Mar=2 ...), write a program to find out total number of days in the given month for the given year. Include a class UserMainCode with a static method “getNumberOfDays” that
accepts 2 integers as arguments and returns an integer. The first argument corresponds to the year and the second argument corresponds to the month code. The method returns an integer corresponding to the number of days in the month.
Create a class Main which would get 2 integers as input and call the static method getNumberOfDays present in the UserMainCode.
Input and Output Format:
Input consists of 2 integers that correspond to the year and month code. Output consists of an integer that corresponds to the number of days in the month in the given year.
Sample Input: Sample Output:
2000 29
1

UserMainCode

```
package javaAssessment;

import java.util.*;

public class UserMainCode {
public static int getNumberOfDays(int y,int c)
{
 Calendar cal=Calendar.getInstance();
 cal.set(Calendar.YEAR, y);
 cal.set(Calendar.MONTH, c);
 int day=cal.getActualMaximum(Calendar.DAY_OF_MONTH);
 return day;
}
}
```

Main

```
package javaAssessment;

import java.util.*;
public class Main {
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
int y=s.nextInt();
int c=s.nextInt();
System.out.println(UserMainCode.getNumberOfDays(y, c));
s.close();
}
}
```

```
2000
1
29
```

2.1.grade calculator (code:53): Pass or Fail

A School wants to give assign grades to its students based on their marks. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:
Read student details from the User. The details would include name, mark in the given order. The datatype for name is string, mark is float.
You decide to build a hashmap. The hashmap contains name as key and mark as value.
BUSINESS RULE:
1. If Mark is less than 60, then grade is FAIL.
2. If Mark is greater than or equal to 60, then grade is PASS.
Note: FAIL/PASS should be in uppercase.
Store the result in a new Hashmap with name as Key and grade as value.
4. You decide to write a function calculateGrade which takes the above hashmap as input and returns the hashmap as output. Include this function in class UserMainCode.
Create a Class Main which would be used to read student details in step 1 and build the hashmap. Call the static method present in UserMainCode.
Input and Output Format:
Input consists of student details. The first number indicates the size of the students. The next two values indicate the name, mark.
Output consists of a name and corresponding grade for each student.
Refer sample output for formatting specifications.
Sample Input 1:
3
Avi
76.36
Sunil
68.42
Raja
36.25
Sample Output 1:
Avi
PASS
Sunil
PASS
Raja
FAIL

UserMainCode

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.TreeMap;
public class UserMainCode {
 public static TreeMap<String, String>calculategrade(HashMap<String, Float>hm){
 TreeMap<String, String> tm = new TreeMap<String, String>();
 Iterator<String> it = hm.keySet().iterator();
 while(it.hasNext()){
 String name = it.next();
 float mark = hm.get(name);
 if(mark >= 60){
```

```
 tm.put(name,"PASS");
 } else if(mark <= 60){
 tm.put(name,"FAIL");
 }
 }
 return tm;
 }
}
```

Main

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;
import java.util.TreeMap;
public class Main {
 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
 float s = scanner.nextFloat();
 scanner.nextLine();
 HashMap<String, Float> hm = new HashMap<String, Float>();
 for(int i = 0; i < s; i++){
 hm.put(scanner.nextLine(), scanner.nextFloat());
 scanner.nextLine();
 }
 TreeMap<String, String> tm = new TreeMap<String, String>();
 tm = UserMainCode.calculategrade(hm);
 Iterator<String> it = tm.keySet().iterator();
 for(int i = 0; i < s; i++){
 String n = it.next();
 String fac = tm.get(n);
 System.out.println(n);
 System.out.println(fac);
 }
 }
}
```

```
Raja
36.35
Avi
PASS
Raja
FAIL
Sunil
PASS
```

2.2.grade calculator (code:28): Gold Silver

A School wants to assign grades to its students based on their marks. You have been assigned as the programmer to
automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the
following steps:
1. Read student details from the User. The details would include roll no, mark in the given order. The datatype for id
is integer, mark is integer.
2. You decide to build a hashmap. The hashmap contains roll no as key and mark as value.
3. BUSINESS RULE:
1. If Mark is greater than or equal to 80 store medal as ""GOLD"".
2. If Mark is less then to 80 and greater than or equal to 60 store medal as ""SILVER"".
3 .If Mark is less then to 60 and greater than or equal to 45 store medal as ""BRONZE"" else store ""FAIL"".
Store the result in TreeMap in which Roll No as Key and grade as value.
4. You decide to write a function calculateGrade which takes the above hashmaps as input and returns the treemap as
output. Include this function in class UserMainCode.
Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static
method present in UserMainCode.
Input and Output Format:
Input consists of employee details. The first number indicates the size of the students. The next two values indicate the roll
id, mark.
Output consists of a single string.
Refer sample output for formatting specifications.
Sample Input 1:
2
1010
80
100
40
Sample Output 1:
100
FAIL
1010
GOLD

UserMainCode

```
import java.util.Iterator;
import java.util.HashMap;
import java.util.TreeMap;
public class UserMainCode
{
public static TreeMap<Integer,String>calculateGrade(HashMap<Integer,Integer>hm)
```

```
{
TreeMap<Integer,String>tm=new TreeMap<Integer,String>();
Iterator<Integer> it=hm.keySet().iterator();
while(it.hasNext())
{
int id=it.next();
int mark=hm.get(id);
if(mark>=80)
tm.put(id,"GOLD");
else if(mark<80 && mark>=60)
tm.put(id,"SILVER");
else if(mark<60 && mark>=45)
tm.put(id,"BRONZE");
else
tm.put(id,"FAIL");
}
return tm;
}}
```

Main

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.TreeMap;
import java.util.Scanner;
public class Main {
public static void main(String []args){
Scanner sc=new Scanner(System.in);
int s=sc.nextInt();
HashMap<Integer,Integer>hm=new HashMap<Integer,Integer>();
for(int i=0;i<s;i++)
{
hm.put(sc.nextInt(),sc.nextInt());
}
TreeMap<Integer,String>tm=new TreeMap<Integer,String>();
tm=UserMainCode.calculateGrade(hm);
Iterator<Integer> it=tm.keySet().iterator();
for(int i=0;i<s;i++)
{
int n=it.next();
String fac=tm.get(n);
System.out.println(n);
System.out.println(fac);
}
}
}
```

```
2
1010
80
100
40
100
FAIL
1010
GOLD
```

3.Sorted array

Write a program to read a string array, remove duplicate elements and sort the array.
Note:
1. The check for duplicate elements must be case-sensitive. (AA and aa are NOT duplicates)
2. While sorting, words starting with upper case letters takes precedence.
Include a class UserMainCode with a static method orderElements which accepts the string
array. The return type is the sorted array.
Create a Class Main which would be used to accept the string arrayand integer 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 string values.
Output consists of the elements of string array.
Refer sample output for formatting specifications.
Sample Input 1:
6
AAA
BBB
AAA
AAA
CCC
CCC
Sample Output 1:
AAA
BBB
CCC
Sample Input 2:
7
AAA
BBB
aaa
AAA
Abc
A
b
Sample Output 2:
A
AAA
Abc
BBB
aaa
b

UserMainCode

```
import java.util.Iterator;
```

```
import java.util.*;
public class UserMainCode {
public static String[] orderElements(String[] arr)
{
HashSet<String> al=new HashSet<String>();
for(int i=0;i<arr.length;i++)
{
al.add(arr[i]);
}
Iterator<String> itr=al.iterator();
String ar[] = new String[al.size()];
int i =0 ;
while(itr.hasNext()){
ar[i] = itr.next();
i++;
}
Arrays.sort(ar);
return ar;
}
}
```

Main

```
import java.util.*;
public class Main {
public static void main(String[] args)
{
int n;
Scanner sin = new Scanner(System.in);
n = sin.nextInt();
String[] a1 = new String[n];
for(int i=0;i<n;i++)
{
a1[i] = sin.next();
}
a1 = UserMainCode.orderElements(a1);
for(int i=0;i<a1.length;i++)
System.out.println(""+a1[i]);
}
}
```

```
6
AAA
BBB
AAA
CCC
AAA
CCC
AAA
BBB
CCC
```

4.1 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

UserMainCode

```
import java.util.Iterator;
import java.util.*;
public class UserMainCode {
public static int sumElements(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;
}
}
```

Main

```
import java.util.*;
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.sumElements(a));
}}
```

```
7
2
3
54
1
6
7
7
62
```

4.2 UniqueEven sum (code 1)

Write a program to read an array, eliminate duplicate elements and calculate the sum of even numbers (values) present in the
array.
Include a class UserMainCode with a static method addUniqueEven which accepts a single integer array. The return type
(integer) should be the sum of the even numbers. In case there is no even number it should return -1.
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.
In case there is no even integer in the input array, print no even numbers as output. Else print the sum.
Refer sample output for formatting specifications.
Assume that the maximum number of elements in the array is 20.
Sample Input 1:
4
2
5
1
4
Sample Output 1:
6
Sample Input 2:
3
1
1
1
Sample Output 2:
no even numbers

UserMainCode;

```
import java.util.Iterator;
import java.util.LinkedHashSet;
public class UserMainCode
{
public static int addUniqueEven(int[] a)
{
int sum=0;
LinkedHashSet<Integer> hm=new LinkedHashSet<Integer>(); for(int i=0;i<a.length;i++
)
{
hm.add(a[i]);
}
Iterator<Integer> im=hm.iterator();
while(im.hasNext())
{
int b=im.next();
if(b%2==0)
sum=sum+b;
}
```

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

Main:

```
import java.util.Scanner;
public class Main
{
public static void main(String[] args)
{
Scanner s=new Scanner(System.in);
int n=s.nextInt();
int[] a=new int [n];
for(int i=0;i<n;i++)
{
a[i]=s.nextInt();
} int t = UserMainCode.addUniqueEven(a);
if (t==-1){
	System.out.println("no even numbers");
}else{
	System.out.println(t);
}
s.close();
}
}
```

```
3
1
1
1
no even numbers
```

5.1word count (Code 85)

Given a string array (s) with each element in the array containing alphabets or digits. Write a program to add all the digits in every string and return the sum as an integer. 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.
Include a class UserMainCode with a static method sumOfDigits which accepts the string array. The return type is the integer formed based on rules. Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.
Input and Output Format: Input consists of a an integer indicating the number of elements in the string array. Output consists of a integer . Refer sample output for formatting specifications.
Sample Input 1: 5 AAA1
B2B 4CCC A5 ABCDE Sample Output 1: 12
Sample Input 2: 3 12 C23 5CR2 Sample Output 2: 15

UserMainCode

```
public class UserMainCode {
 public static int sumOfDigits(String[] s1)
 {
 int sum = 0;
 for(int i=0;i<s1.length;i++)
 {
 String s = s1[i];
 for(int j = 0;j<s.length();j++)
 {
 Character c = s.charAt(j);
 if(Character.isDigit(c))
 {
 sum+=Integer.parseInt(s.valueOf(c));
 }
 }
 }
 return sum;
 }
}
```

Main class

```
import java.util.Scanner;
public class Main {
public static void main(String[] args)
 {
 Scanner s=new Scanner(System.in);
 int n=s.nextInt();
 String a[]=new String[n];
 for(int i=0;i<n;i++)
 {
 a[i]=s.next();
 }
 System.out.println(UserMainCode.sumOfDigits(a));
```

```
 }
}
```

5.2 Word Count

Given a string array (s) and non negative integer (n) and return the number of elements in the array which have same number of characters as the givent int N.
Include a class UserMainCode with a static method countWord which accepts the string array and integer. The return type is the string formed based on rules.
Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.
Input and Output Format:
Input consists of a an integer indicating the number of elements in the string array followed the elements and ended by the non-negative integer (N).
Output consists of a integer .
Refer sample output for formatting specifications.
Sample Input 1:
4
a
bb
b
ccc
1
Sample Output 1:
2
Sample Input 2:
5
dog
cat
monkey
bear
fox
3
Sample Output 2:
3

UserMainCode:

```
public class UserMainCode {
public static int countWord(int n,String[] arr,int a)
 {
 int count=0;
 for(int i=0;i<arr.length;i++)
 {
 if(arr[i].length()==a)
 {
 count++;
 }
 }
 return count;
 }
}
```

Main:

```
import java.util.*;
```

```
public class Main {
public static void main(String []args){
 Scanner sc=new Scanner(System.in);
 int n=sc.nextInt();
 String[] arr=new String[n];
 for(int i=0;i<n;i++)
 {
 arr[i]=sc.next();
 }
 int a=sc.nextInt();
 System.out.println(UserMainCode.countWord(n,arr,a));
 sc.close();
}}
```

```
4
a
bb
b
ccc
1
2
```

5.3 Word Count - II

Write a program to read a string and count the number of words present in it.
Include a class UserMainCode with a static method countWord which accepts the string.
The return type is the integer giving out the count of words.
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:
Today is Sunday
Sample Output 1:
3

UserMainCode:

```
import java.util.StringTokenizer;
public class UserMainCode {
public static void countWord(String s1){
 StringTokenizer st=new StringTokenizer(s1," ");
 int n=st.countTokens();
 System.out.println(n);
}
}
```

Main class:

```
import java.util.*;
public class Main {
 public static void main(String[] args) {
 Scanner s=new Scanner(System.in);
 String s1=s.nextLine();
```

```
UserMainCode.countWord(s1);
s.close();
}
}
```

6.1.dateofbirth

Write a program to validate the Date of Birth given as input in String format (MM/dd/yyyy) as per the validation rules given below. Return true for valid dates else return false.
1. Value should not be null
2. month should be between 1-12, date should be between 1-31 and year should be a four digit number.
Include a class UserMainCode with a static method ValidateDOB which accepts the string. The return type is 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:
12/23/1985
Sample Output 1:
TRUE
Sample Input 2:
31/12/1985
Sample Output 2:
FALSE

UserMainCode

```
import java.text.SimpleDateFormat;
import java.util.Date;
public class UserMainCode {
public static Boolean ValidateDOB(String str){
 Boolean b=false;
 SimpleDateFormat sdf=new SimpleDateFormat("MM/dd/yyyy");
 sdf.setLenient(false);
 try
 {
 Date d1=sdf.parse(str);
 return b=true;
 }
 catch(Exception e)
 {
 return b=false;
 }
}
}
```

Main

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

```
String str=new String();
Scanner sc=new Scanner(System.in);
str=sc.nextLine();
Boolean b=UserMainCode.ValidateDOB(str);
if(b==true)
 System.out.println("TRUE");
if(b==false)
 System.out.println("FALSE");
}
}
```

```
12/23/1985
TRUE
```

6.2.dayofbirth

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

UserMainCode

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.*;
public class UserMainCode {
 public static String calculatebornday(String input) throws ParseException{
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd-MM-yyyy");
 SimpleDateFormat simpleDateFormat1 = new SimpleDateFormat("EEEEE");
 Date d = simpleDateFormat.parse(input);
 String s1 = simpleDateFormat1.format(d);
 String s2 = s1.toUpperCase();
 return s2;
 }
```

Main

```
import java.text.ParseException;
import java.util.*;
public class Main {
public static void main(String[] args) throws ParseException {
 Scanner scanner = new Scanner(System.in);
 String input = scanner.nextLine();
 System.out.println(UserMainCode.calculatebornday(input));
 }
}
```

```
29-07-2013
MONDAY
```