21. max Scorer

Write a program that performs the following actions:

1. Read n strings as input and stores them as an arraylist. The string consists of student information like name and obtained marks of three subjects. Eg: name-mark1-mark2-mark3 [suresh-70-47-12] The mark would range between 0 – 100 (inclusive).
2. Write a function **highestScorer** which accepts these the arraylist and returns the name of the student who has scored the max marks. Assume the result will have only one student with max mark.

Include a class UserMainCode with the static method **highestScorer** which accepts the arraylist and returns the name (string) of max scorer.

Create a Class Main which would be used to read n strings into arraylist and call the static method present in UserMainCode.

Input and Output Format:

Input consists of 1 integer and n strings. The first integer denotes the size of the arraylist, the next n strings are score pattern described above.

Output consists of a string with the name of the top scorer.

Refer sample output for formatting specifications.

Sample Input 1:

3

sunil-56-88-23

bindul-88-70-10

john-70-49-65

Sample Output 1:

john

Solution:

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
```

```
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
ArrayList<String> a=new ArrayList<String>();
for(int i=0;i<n;i++)
a.add(sc.next());
System.out.println(User.highestScorer(a));
sc.close();
}
}
```

```
import java.util.ArrayList;
import java.util.StringTokenizer;

public class User {
public static String highestScorer(ArrayList<String> a) {
String ss=null,name=null,Name=null;
int m1=0,m2=0,m3=0,sum=0,max=0;
for(int i=0;i<a.size();i++)
{
ss=a.get(i);
StringTokenizer st=new StringTokenizer(ss,"-");
while(st.hasMoreTokens())
{
name=st.nextToken();
m1=Integer.parseInt(st.nextToken());
m2=Integer.parseInt(st.nextToken());
m3=Integer.parseInt(st.nextToken());
sum=m1+m2+m3;
if(max<sum)
{
max=sum;
Name=name;
}
}
}
return Name;
}
}
```

22. Max Vowels

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

Include a class UserMainCode with a static method **getWordWithMaximumVowels** which accepts a string The return type is the longest word of type 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 a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

Appreciation is the best way to motivate

Sample Output 1:

Appreciation

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.nextLine();
System.out.println(User.getWordWithMaximumVowels(s));
}
}


import java.util.StringTokenizer;

publicclass User {
publicstatic String getWordWithMaximumVowels(String s) {
StringTokenizer st = new StringTokenizer(s, " ");
int count = 0, max = 0;
String res = null;
String f = null;
while (st.hasMoreTokens()) {
res = st.nextToken();
count = 0;
for (int k = 0; k < res.length(); k++) {
if (res.charAt(k) == 'a' || res.charAt(k) == 'e'
|| res.charAt(k) == 'i' || res.charAt(k) == 'o'
|| res.charAt(k) == 'u' || res.charAt(k) == 'A'
|| res.charAt(k) == 'E' || res.charAt(k) == 'I'
|| res.charAt(k) == 'O' || res.charAt(k) == 'U')
count++;
if (count > max) {
max = count;
f = res;
}

}
}
return f;
}
}
```

23. All Vowels

Write a Program to check if given word contains exactly five vowels and the vowels are in alphabetical order. Return 1 if the condition is satisfied else return -1. Assume there is no repetition of any vowel in the given string and all letters are in lower case.

Include a class UserMainCode with a static method **testOrderVowels** which accepts a string The return type is integer based on the condition stated above.

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 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

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.nextLine();
int res = User.testOrderVowels(s);
if (res == 1)
System.out.println("valid");
else
System.out.println("invalid");
}
}


publicclass User {
publicstaticint testOrderVowels(String s1) {

StringBuffer sb = newStringBuffer();
int res = 0;
for (int i = 0; i < s1.length(); i++) {
if (s1.charAt(i) == 'a' || s1.charAt(i) == 'A'
|| s1.charAt(i) == 'e' || s1.charAt(i) == 'E'
|| s1.charAt(i) == 'i' || s1.charAt(i) == 'I'
|| s1.charAt(i) == 'o' || s1.charAt(i) == 'O'
|| s1.charAt(i) == 'u' || s1.charAt(i) == 'U') {
sb.append(s1.charAt(i));
}
}
if (sb.toString().equals("aeiou"))
res = 1;
else
res = 0;
return res;
}
}
```

24. Adjacent Swaps

Write a Program that accepts a string as a parameter and returns the string with each pair of adjacent letters reversed. If the string has an odd number of letters, the last letter is unchanged.

Include a class UserMainCode with a static method **swapPairs** which accepts a string. The return type is string which is reversed pair of letters.

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 a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

forget

Sample Output 1:

ofgrte

Sample Input 2:

New York

Sample Output 2:

eN woYkr

```
import java.util.Scanner;

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

```
Scanner sc = new Scanner(System.in);
String string=sc.nextLine();
System.out.println(User.swapPairs(string));
sc.close();
}
}


public class User {
public static String swapPairs(String s) {
StringBuffer sb=new StringBuffer();
if(s.length()%2==0)
{
for(int i=0;i<s.length()-1;i=i+2)
{
sb.append(s.charAt(i+1)).append(s.charAt(i));
}
}
else
{
for(int i=0;i<s.length()-1;i=i+2)
{
sb.append(s.charAt(i+1)).append(s.charAt(i));
}
sb.append(s.charAt(s.length()-1));
}
return sb.toString();
}
}
```

25. Sum of Digits

Write a Program that accepts a word as a parameter, extracts the digits within the string and returns its sum.

Include a class UserMainCode with a static method **getdigits** which accepts a string. The return type is integer representing the sum.

Create a Class Main which would be used to accept the 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 single string.

Refer sample output for formatting specifications.

Sample Input 1:

abc12de4

Sample Output 1:

7

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.next();
System.out.println(User.getdigits(s));
}
}


publicclass User {
publicstaticint getdigits(String s) {
int sum = 0, n = 0;
for (int i = 0; i < s.length(); i++) {
if (s.charAt(i) >= 65 && s.charAt(i) <= 90 || s.charAt(i) >= 97
&&s.charAt(i) <= 122)
;
else {
n = Character.getNumericValue(s.charAt(i));
sum = sum + n;
}
}
return sum;
}
}
```

```
        public static String removeDuplicate(String s) {

                int sum = 0,n=0;
                for (int i = 0; i < s.length(); i++) {
                        if( Character.isDigit(s.charAt(i))  )
                        {
                               int c=Character.getNumericValue(s.charAt(i));
                        sum = sum +c;

                        }
                  }
                        String sum1=String.valueOf(sum);
                        return sum1;
}
```

26. Password

Given a String , write a program to find whether it is a valid password or not.

Validation Rule:
Atleast 8 characters
Atleast 1 number(1,2,3...)
Atleast 1 special character(@,#,%...)
Atleast 1 alphabet(a,B...)

Include a class **UserMainCode** with a static method "**validatePassword**" that accepts a String argument and returns a boolean value. The method returns true if the password is acceptable. Else the method returns false.
Create a class **Main** which would get a String as input and call the static method **validatePassword** present in the UserMainCode.

Input and Output Format:
Input consists of a String.
Output consists of a String that is either "Valid" or "Invalid".

Sample Input 1:
cts@1010

Sample Output 1:
Valid

Sample Input 2:
punitha3

Sample Output 2:
Invalid

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.next();
boolean flag = User.validatePassword(s);
if (flag == true)
System.out.println("valid");
else
System.out.println("invalid");
}
}

publicclass User {
publicstaticboolean validatePassword(String s) {
int number = 0, c = 0, sp = 0;
boolean flag = false;
for (int i = 0; i < s.length(); i++) {
if (s.length() >= 8) {
if (Character.isDigit(s.charAt(i))) {
number++;
}
if (Character.isLetter(s.charAt(i))) {
c++;
} else {
if (s.charAt(i) != ' '&& !Character.isDigit(s.charAt(i))
&& !Character.isLetter(s.charAt(i)))
sp++;
}
}
}
if (number >= 1 && c >= 1 && sp >= 1)
flag = true;
return flag;
}
}
```

```
public static boolean removeDuplicate(String s) {

                    int number = 0, c = 0, sp = 0,len=0;
                    boolean flag = false;
                    for (int i = 0; i < s.length(); i++)
                    {
                    if (s.length() >= 8)
                    {
                          len++;
                    }
                    if (Character.isDigit(s.charAt(i)))
                    {
                    number++;
                    }
                    if (Character.isLetter(s.charAt(i)))
                    {
                    c++;
                    }

                    if (s.charAt(i) != ' '&& !Character.isDigit(s.charAt(i))
                    && !Character.isLetter(s.charAt(i)))
                    {
                    sp++;
                    }
                    }

                    if (number >= 1 && c >= 1 && sp >= 1 && len>1)
                    flag = true;

                    return flag;

}
```

27. Employee Bonus

A Company wants to give away bonus to its employees. 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 Employee details from the User. The details would include id, DOB (date of birth) and salary in the given order. The datatype for id is integer, DOB is string and salary is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and salary as value.
3. If the age of the employee in the range of 25 to 30 years (inclusive), the employee should get bonus of 20% of his salary and in the range of 31 to 60 years (inclusive) should get 30% of his salary. store the result in TreeMap in which Employee ID as key and revised

salary as value. Assume the age is caculated based on the date 01-09-2014. (Typecast the bonus to integer).

4. Other Rules:

   a. If Salary is less than 5000 store -100.

   b. If the age is less than 25 or greater than 60 store -200.

c. a takes more priority than b i.e both if a and b are true then store -100.

5. You decide to write a function **calculateRevisedSalary** 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 employees. The next three values indicate the employee id, employee DOB and employee salary. The Employee DOB format is “dd-mm-yyyy”

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

2

1010

20-12-1987

10000

2020

01-01-1985

14400

Sample Output 1:

1010

12000

2020

17280

Solution:

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

public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
TreeMap<Integer,Integer> t=new TreeMap<Integer,Integer>();
HashMap<Integer,String> h1=new HashMap<Integer,String>();
HashMap<Integer,Integer> h2=new HashMap<Integer,Integer>();
for(int i=0;i<n;i++)
{
       int id=sc.nextInt();
       h1.put(id, sc.next());
       h2.put(id, sc.nextInt());
}
t=User.calSalary(h1,h2);
Iterator<Integer> it1=t.keySet().iterator();
while(it1.hasNext())
{
       int id=it1.next();
       int val=t.get(id);
       System.out.println(id);
       System.out.println(val);
}
sc.close();
}
}
```

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

public class User {
public static TreeMap<Integer,Integer> calSalary(HashMap<Integer,String> h1,
HashMap<Integer,Integer> h2) throws ParseException {
       TreeMap<Integer,Integer> t=new TreeMap<Integer,Integer>();
       Iterator<Integer> it1=h1.keySet().iterator();
       SimpleDateFormat sd=new SimpleDateFormat("dd-MM-yyyy");
       String ss="01-09-2014";
       int new_sal=0;
       while(it1.hasNext())
       {
              int id1=it1.next();
              String dob=h1.get(id1);
              int salary=h2.get(id1);
```

```
                Date d1=sd.parse(dob);
                Date d2=sd.parse(ss);
                d1=sd.parse(dob);
                int y1=d1.~~getYear~~();
                int y2=d2.~~getYear~~();
                int year=Math.abs(y1-y2);
                if(year>=25 && year<=30)
                {
                        new_sal=salary+(salary*20/100);
                        t.put(id1,new_sal);
                }
                else if(year>=31 && year<=60)
                {
                        new_sal=salary+(salary*30/100);
                        t.put(id1,new_sal);
                }
                else
                        ;

        }
        return t;
}
}
```

28. Grade Calculator

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
```

Solution:

```
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 sc = new Scanner(System.in);
```

```
int n = sc.nextInt();

int i;

HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();

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

hm.put(sc.nextInt(), sc.nextInt());

}

TreeMap<Integer, String> t = new TreeMap<Integer, String>();

t.putAll(User.display(n, hm));

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

while (it.hasNext()) {

int r = it.next();

String g = t.get(r);

System.out.println(r);

System.out.println(g);

}}}

import java.util.HashMap;
import java.util.Iterator;
import java.util.TreeMap;

publicclass User {

publicstatic TreeMap<Integer, String> display(int n,
HashMap<Integer, Integer> h) {
TreeMap<Integer, String> t = new TreeMap<Integer, String>();
Iterator<Integer> i = h.keySet().iterator();
while (i.hasNext()) {
int r = i.next();
int m = h.get(r);
if (m >= 80)
t.put(r, "GOLD");
elseif (m < 80 && m >= 60)
t.put(r, "SILVER");
elseif (m < 60 && m >= 45)
t.put(r, "BRONZE");
else
t.put(r, "FAIL");
}
return t;

}
}
```

29. DigitSum

Write a program to read a non-negative integer n, compute the sum of its digits. If sum is greater than 9 repeat the process and calculate the sum once again until the final sum comes to single digit.Return the single digit.
Include a class UserMainCode with a static method **getDigitSum** which accepts the integer value. The return type is 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 integer.
Output consists of integer.
Refer sample output for formatting specifications.

Sample Input 1:
9999
Sample Output 1:
9

Sample Input 2:
698
Sample Output 2:
5

Solution:

```
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
int n = s.nextInt();
System.out.println(User.getDigitSum(n));
}

}


publicclass User {
publicstaticint getDigitSum(int n) {
int sum = 0;
while (n > 10) {
int r = 0;
sum = 0;
while (n != 0) {
r = n % 10;
sum = sum + r;
n = n / 10;
}
n = sum;
}

return sum;
}
}
```

30. Anagrams

Write a program to read two strings and checks if one is an anagram of the other.
An 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'.
Include a class UserMainCode with a static method **checkAnagram** which accepts the two strings. The return type is boolean which is TRUE / FALSE.
Create a Class Main which would be used to accept the two strings and call the static method present in UserMainCode.

Input and Output Format:
Input consists of two strings.
Output consists of TRUE / FALSE.
Refer 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

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s1 = sc.nextLine();
String s2 = sc.nextLine();
boolean b = User.checkAnagram(s1, s2);
if (b == true)
System.out.println("TRUE");
else
System.out.println("FALSE");
}
}
```

```
import java.util.ArrayList;
import java.util.Collections;

publicclass User {
publicstaticboolean checkAnagram(String s1, String s2) {
boolean b = false;
ArrayList<Character> a1 = new ArrayList<Character>();
ArrayList<Character> a2 = new ArrayList<Character>();
ArrayList<Character> a3 = new ArrayList<Character>();
for (int i = 0; i < s1.length(); i++)
a1.add(s1.toLowerCase().charAt(i));
for (int i = 0; i < s2.length(); i++)
a2.add(s2.toLowerCase().charAt(i));
a3.add(' ');
a1.removeAll(a3);
a2.removeAll(a3);
Collections.sort(a1);
Collections.sort(a2);
if (a1.equals(a2))
b = true;
return b;
}
}


public static void main(String[] args) {

            Scanner sc = new Scanner(System.in);
            String s1 = sc.nextLine();
            String s2 = sc.nextLine();
            boolean b =Anagrams.check(s1, s2);
            if (b == true)
            System.out.println("TRUE");
            else
            System.out.println("FALSE");

      }
}




      public class Anagrams
      {
            public static boolean check(String s1,String s2)
            {
            boolean res=false;
                  ArrayList<Character> a1=new ArrayList<Character>();
                  ArrayList<Character> a2=new ArrayList<Character>();
                  for(int i=0;i<s1.length();i++)
                  {
                        a1.add(s1.charAt(i));
                  }

                  for(int i=0;i<s2.length();i++)
                  {
                        a2.add(s2.charAt(i));
                  }
                  Collections.sort(a1);
                  Collections.sort(a2);


                  if((a1.containsAll(a2))|| (a2.containsAll(a1)))
                  {
                        res=true;
                  }
                  return res;
            }
      }
```