41)Calculate Average – Hash Map

Write amethod that accepts the input data as a hash map and finds out the avg of all values whose keys are odd numbers.

Include a class **UserMainCode** with a static method **calculateAverage** which accepts a HashMap<Integer,Double> and the size of the HashMap. The return type (Double) should return the calculated average. Round the average to two decimal places and return it.

Create a Class Main which would be used to accept Input values and store it as a hash map, and call the static method present in UserMainCode.

Input and Output Format:

Input consists of an integer n corresponds to number of hash map values, followed by 2n values. (index followed by value).

Output consists of a Double.

Refer sample input and output for formatting specifications.

Sample Input :
4
1
3.41
2
4.1
3
1.61
4
2.5
Sample Output :
2.51

42) Count Sequential Characters

109.Get a string as input and write code to count the number of characters which gets repeated 3 times consecutively and return that count (ignore case). If no character gets repeated 3 times consecutively return -1.

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

The return type of the output is the repeat count.

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

Input and Output Format:

Input consists a string.

Output is an integer.

Refer sample output for formatting specifications.

Sample Input 1:

abcXXXabc

Sample Output 1:

1

Sample Input 2:

aaaxxyzAAAx

Sample Output 2:

2

```
public class Main {

public static void main(String[] args) {

String input1="aaxxyzAAx";

System.out.println(consecutiveRepeatitionOfChar(input1));

}


public static int consecutiveRepeatitionOfChar(String input1) {

int c=0;

int n=0;
```

```
for(int i=0;i<input1.length()-1;i++){

if(input1.charAt(i)==input1.charAt(i+1))

n++;

else

n=0;

if(n==2)

c++; }

return c;

}

}
```

43) Length of the Largest Chunk

Write a program to read a string and find the length of the largest chunk in the string. If there are no chunk print “No chunks” else print the length.
NOTE: chunk is the letter which is repeating 2 or more than 2 times.
Include a class **UserMainCode** with a static method **largestChunk** which accepts a string. The return type (Integer) should return the length of the largest chunk if the chunk is present, else return -1.

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.

Refer sample output for formatting specifications.

Sample Input 1:
You are toooo good
Sample Output 1:
4
(Because the largest chunk is letter 'o' which is repeating 4 times)

Sample Input 2:
who are u

Sample Output 2:
No chunks

```
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);

}
```

}

44) Unique Characters in a string

Write a program that takes a string and returns the number of unique characters in the string. If the given string doest not contain any unique characters return -1

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

The return type of the output is the count of all unique characters in the strings.

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

Input and Output Format:

Input consists a string.

Output is an integer.

Refer sample output for formatting specifications.

Sample Input 1:

HelloWorld

Sample Output 1:

5

Sample Input 2:

coco

Sample Output 2:

-1

```
public class Main {

public static void main(String[] args) {

String s1="HelloWorld";
```

```
getvalues(s1);
}
public static void getvalues(String s1) {
String s2=s1.toLowerCase();
StringBuffer sb=new StringBuffer(s2);
int l=sb.length();
int count=0;
for(int i=0;i<l;i++)
{ count=0;
for(int j=i+1;j<l;j++)
{
if(sb.charAt(i)==sb.charAt(j))
{
sb.deleteCharAt(j);
count++;
j--;
l--;
j=i;
}
}
if(count>0)
{
```

```
sb.deleteCharAt(i);

i--;

l--;

}

}

if(sb.length()==0)

{

System.out.println(-1);

}

else

System.out.println(sb.length());

}

}
```

45) Name Shrinking

Write a program that accepts a string as input and converts the first two names into dot-separated initials and printa the output.

Input string format is 'fn mn ln'. Output string format is 'ln [mn's 1st character].[fn's 1st character]'

Include a class **UserMainCode** with a static method **getFormatedString** which accepts a string. The return type (String) should return the shrinked name.

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:
Sachin Ramesh Tendulkar
Sample Output:
Tendulkar R.S

```
import java.util.Scanner;
public class Main {


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
System.out.println(UserMainCode.getFormatedString(s));



}
}
```

```
import java.util.StringTokenizer;


public class UserMainCode {
	public static String getFormatedString(String s)

	{
		StringTokenizer st=new StringTokenizer(s," ");
		StringBuffer sb=new StringBuffer();
		while(st.hasMoreTokens())
		{
		String a=st.nextToken();
		String b=st.nextToken();
		String c=st.nextToken();
		sb.append(c.substring(0));
		sb.append(" ");
		sb.append(b.substring(0,1));
		sb.append(".");
		sb.append(a.substring(0,1));
		//String ss=sb.toString();
		}
		return sb.toString();
	}
```

46) Odd Digit Sum

Write a program to input a String array. The input may contain digits and alphabets (“de5g4G7R”). Extract odd digits from each string and find the sum and print the output.
For example, if the string is "AKj375A" then take 3+7+5=15 and not as 375 as digit.
Include a class **UserMainCode** with a static method **oddDigitSum** which accepts a string array and the size of the array. The return type (Integer) should return the sum.

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

Assume maximum length of array is 20.

Input and Output Format:

Input consists of an integer n, corresponds to the number of strings, followed by n Strings.

Output consists of an Integer.

Refer sample output for formatting specifications.

Sample Input :
3
cog2nizant1
al33k
d2t4H3r5
Sample Output :
15
(1+3+3+3+5)

```
import java.util.Scanner;


  public class kape {

public static void main(String[] args) {

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

```
        int s1=sc.nextInt();
        String[] s2 = new String[s1];
        for (int i = 0; i < s1; i++) {
        s2[i] = sc.next();
        }
System.out.println(kape1.getSum(s2));
}}

public class kape1 {

public static int getSum(String[] s1) {
int sum=0;
for(int i=0;i<s1.length;i++)
for(int j=0;j<s1[i].length();j++){
char c=s1[i].charAt(j);
if(Character.isDigit(c)){
if(c%2!=0)
{
String t=String.valueOf(c);
int n=Integer.parseInt(t);
sum=sum+n; } }}
return sum;
```

}

}

47) Unique Number

Write a program that accepts an Integer as input and finds whether the number is Unique or not. Print Unique if the number is “Unique”, else print “Not Unique”.

Note: A Unique number is a positive integer (without leading zeros) with no duplicate digits.For example 7, 135, 214 are all unique numbers whereas 33, 3121, 300 are not.

Include a class **UserMainCode** with a static method **getUnique** which accepts an integer. The return type (Integer) should return 1 if the number is unique else return -1.

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

Input and Output Format:

Input consists of an integer .

Output consists of a String (“Unique” or “Not Unique”).

Refer sample output for formatting specifications.

Sample Input 1:
123
Sample Output 1:
Unique

Sample Input 2:
33
Sample Output 2:
Not Unique

```
public class useer{

    public static void main(String[]args)

    {

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

```
int n=sc.nextInt();
int []a=new int[100];
int i=0,count=0;
while(n!=0)
{
	int num=n%10;
	a[i]=num;
	i++;
	n=n/10;
}
for(int j=0;j<i-1;j++)
{
	for(int k=j+1;k<=i-1;k++)
	{
		if(a[j]==a[k]){
			count++;
		}
	}}
		if(count>0)
		{
			System.out.println("Invalid/not unique");
		}
```

```
else
{
        System.out.println("valid/unique");
}
}}
```

48) Sum of Lowest marks

Given input as HashMap, value consists of marks and rollno as key.Find the sum of the lowest three subject marks from the HashMap.

Include a class **UserMainCode** with a static method **getLowest** which accepts a Hashmap with marks and rollno.

The return type of the output is the sum of lowest three subject marks.

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

Input and Output Format:

First line of the input corresponds to the HashMap size.

Input consists a HashMap with marks and rollno.

Output is an integer which is the sum of lowest three subject marks.

Refer sample output for formatting specifications.

Sample Input 1:

```
5
1
54
2
85
3
74
4
59
5
57
```

Sample Output 1:

170

Sample Input 2:

4
10
56
20
58
30
87
40
54

Sample Output 2:

168

49) Color Code Validation

Give a String as colour code as input and write code to validate whether the given string is a valid color code or not.

Validation Rule:
String should start with the Character '#'.
Length of String is 7.
It should contain 6 Characters after '#' Symbol.
It should contain Characters between 'A-F' and Digits '0-9'.
If String acceptable the return true otherwise false.

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

The return type of the output is a boolean which returns true if its is a valid color code else it returns false.

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

Input and Output Format:

Input consists a string corresponding to the color code.

Output is a boolean which returns true or false

Refer sample output for formatting specifications.

Sample Input 1:

#99FF33

Sample Output 1:

true

Sample Input 2:

#CCCC99#

Sample Output 2:

False

```
import java.util.Scanner;

class Main
{
	public static void main(String[] a)
	{
		Scanner sc=new Scanner(System.in);
		String s=sc.nextLine();
		if(s.matches("(#)[A-Z0-9]{6}"))
		{
			System.out.println("valid");
		}
		else
			System.out.println("invalid");

	}
}
```

50) Repeating set of characters in a string

Get a string and a positive integer n as input .The last n characters should repeat the number of times given as second input.Write code to repeat the set of character from the given string.

Include a class **UserMainCode** with a static method **getString** which accepts a string and an integer n as input.

The return type of the output is a string with repeated n characters.

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

Input and Output Format:

Input consists a string and a positive integer n.

Output is a string with repeated characters.

Refer sample output for formatting specifications.

Sample Input 1:

Cognizant
3

Sample Output 1:

Cognizantantantant

Sample Input 2:

myacademy
2

Sample Output 2:

Myacademymymy

```
import java.util.*;

public class useerm {

	public static String lengthiestString(String s1,int n){

		StringBuffer sb=new StringBuffer();

		sb.append(s1);

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


		{


			sb.append(s1.substring(s1.length()-n,s1.length()));

// sb.append(s1.substring(s1.length()-n))
```

```
        }
    return sb.toString();
        }
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
                System.out.println("enter the String:");
                String s1=s.nextLine();
                int n=s.nextInt();
                System.out.println("the lengthiest string is:"+lengthiestString(s1,n));
                }
}
```