**31. Triplets**

Given an integer array, Write a program to find if the array has any triplets. A triplet is a value if it appears 3 consecutive times in the array.

Include a class UserMainCode with a static method **checkTripplets** which accepts an integer array. The return type is boolean stating whether its a triplet or not.

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

**Input and Output Format:**

Input consists of n+1 integers. The first integer would represent the size of array and the next n integers would have the values.

Output consists of a string stating TRUE or FALSE.

Refer sample output for formatting specifications.

**Sample Input 1:**
7
3
3
5
5
5
2
3

**Sample Output 1:**
TRUE

**Sample Input 2:**
7
5
3
5

1
5
2
3

**Sample Output 2:**
FALSE

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

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();

int[] a=new int[n];

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

        a[i]=sc.nextInt();

boolean b=User.checkTripplets(a);

System.out.println(b);

}

}
```

import java.text.ParseException;

```
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;



public class User {
public static boolean checkTripplets  (int a[]) {
boolean b=false;
int c=0;
for(int i=0;i<a.length-2;i++)
{
        if(a[i]==a[i+1]&&a[i+1]==a[i+2])
                b=true;
                else
                        b=false;
}
return b;
}
}
```

**32. Repeat Front**

Given a string (s) and non negative integer (n) apply the following rules.

1. Display the first three characters as front.
2. If the length of the string is less than 3, then consider the entire string as front and repeat it n times.

Include a class UserMainCode with a static method **repeatFirstThreeCharacters** which accepts the string 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 string and integer.

Output consists of a string .

Refer sample output for formatting specifications.

**Sample Input 1:**
Coward
2

**Sample Output 1:**
CowCow

**Sample Input 2:**
So
3

**Sample Output 2:**
SoSoSo

```
import java.text.ParseException;
```

```
import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;


public class Main {

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

Scanner sc = new Scanner(System.in);


String s=sc.next();

int n=sc.nextInt();

String res=User.repeatFirstThreeCharacters(s,n);

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

        System.out.print(res);


}

}
```

```
import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Date;
```

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;



public class User {
public static String repeatFirstThreeCharacters(String s, int n) {

	String front=null;
	if(s.length()>=3)
	{
		front=s.substring(0,3);

	}
	else
		front=s;
	return front;
}
}
```

**33. 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

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

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();

String[] a= new String[n];

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

        a[i]=sc.next();
```

```
String res[]=User.orderElements(a);
for(int i=0;i<res.length;i++)
        System.out.println(res[i]);
}
}
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.StringTokenizer;


public class User {
public static String[] orderElements(String[] s) {

        LinkedHashSet<String> lhs=new LinkedHashSet<String>();
```

```
        for(int i=0;i<s.length;i++)
        {
        lhs.add(s[i]);
        }
        String[] a= new String[lhs.size()];
        for(int i=0;i<s.length;i++)
        {
                lhs.toArray(a);
        }
        Arrays.sort(a);
        return a;
}
}
```

**34. Pattern Matcher**

Write a program to read a string and check if it complies to the pattern 'CPT-XXXXXX' where XXXXXX is a 6 digit number. If the pattern is followed, then print TRUE else print FALSE.

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

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 should print TRUE or FALSE .

Refer sample output for formatting specifications.

**Sample Input 1:**
CPT-302020

**Sample Output 1:**
TRUE

**Sample Input 2:**
CPT123412

**Sample Output 2:**
FALSE

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;


public class Main {

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

Scanner sc = new Scanner(System.in);

String a= sc.next();


boolean b=User.CheckID(a);


        System.out.println(b);

}

}
```

```
import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Calendar;

import java.util.Collections;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashSet;

import java.util.StringTokenizer;



public class User {

public static boolean CheckID (String s) {

boolean b=false;

if(s.matches("(CPT-)[0-9]{6}"))

                    b=true;

else

          b=false;

return b;

}

}
```

**35. Playing with String - I**

Given a string array and non negative integer (n) apply the following rules.

1. Pick nth character from each String element in the String array and form a new String.
2. If nth character not available in a particular String in the array consider $ as the character.
3. Return the newly formed string.

Include a class UserMainCode with a static method **formString** which accepts the string 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 which denotes the size of the array followed by the array of strings and an integer (n).
Output consists of a string .
Refer sample output for formatting specifications.

**Sample Input 1:**
4
ABC
XYZ
EFG
MN
3

**Sample Output 1:**
CZG$

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;
```

```
import java.util.Iterator;

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();

String[] a=new String[n];

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

        a[i]=sc.next();

int s=sc.nextInt();

System.out.println(User.formString(a,s));

}

}


import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Calendar;

import java.util.Collections;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;
```

```
import java.util.LinkedHashSet;
import java.util.StringTokenizer;



public class User {
public static String formString(String s[],int n) {
	StringBuffer sb=new StringBuffer();
	for(int i=0;i<s.length;i++)
	{
		String st=s[i];
		if(st.length()>=n)
		{
			sb.append(st.charAt(n-1));
		}
		else
			sb.append("$");
	}
return sb.toString();
}
}
```

**36. Regular Expression - 1**

Given a string (s) apply the following rules.

1. String should be only four characters long.
2. First character can be an alphabet or digit.
3. Second character must be uppercase 'R'.
4. Third character must be a number between 0-9.

If all the conditions are satisifed then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validate** which accepts the string. The return type is the boolean formed based on rules.

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 or FALSE .

Refer sample output for formatting specifications.

**Sample Input 1:**
vR4u

**Sample Output 1:**
TRUE

**Sample Input 2:**
vRau

**Sample Output 2:**
FALSE

**Sample Input 3:**

vrau

**Sample Output 3:**
FALSE

```
import java.util.Scanner;


public class Main {

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

Scanner sc = new Scanner(System.in);


String s=sc.next();

System.out.println(User.validate (s));


}

}


public class User {

public static boolean validate (String s) {

        boolean b= false;

        if(s.length()==4)

        {

        if(s.matches("[a-z0-9]{1}(R)[0-9]{1}[A-Za-z0-9]{1}"))

                b=true;

        else

                b=false;
```

```
}
        return b;
}
}
```

**37. Regular Expression – 2 (Age Validator)**

Given the age of a person as string, validate the age based on the following rules.

1. Value should contain only numbers.
2. Value should be non-negative.
3. Value should be in the range of 21 to 45'.

If all the conditions are satisifed then print TRUE else print FALSE.

Include a class UserMainCode with a static method **ValidateAge** which accepts the string. The return type is the boolean formed based on rules.

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 or FALSE .
Refer sample output for formatting specifications.

**Sample Input 1:**
23

**Sample Output 1:**
TRUE

**Sample Input 2:**
-34

**Sample Output 2:**
FALSE

**Sample Input 3:**
3a

**Sample Output 3:**
FALSE

```
import java.util.Scanner;


public class Main {

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

Scanner sc = new Scanner(System.in);


int s=sc.nextInt();

System.out.println(User.validate (s));


}

}


public class User {

public static boolean validate (int s) {

        boolean b= false;

        if(s>0)

        {

        if(s>=21&&s<=45)
```

```
                b=true;
        else
                b=false;
}
        return b;
}
}
```

**38. Regular Expression – 3 (Phone Validator)**

Given a phone number as string, validate the same based on the following rules.

1. Value should contain only numbers.
2. Value should contain 10 digits.
3. Value should not start with 00.

If all the conditions are satisifed then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validatePhone** which accepts the string. The return type is the boolean formed based on rules.

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 or FALSE .
Refer sample output for formatting specifications.

**Sample Input 1:**
9987684321

**Sample Output 1:**

TRUE

**Sample Input 2:**
0014623452

**Sample Output 2:**
FALSE

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;


public class Main {

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

Scanner sc = new Scanner(System.in);


String s=sc.next();

System.out.println(User.validatePhone(s));


}

}


public class User {

public static boolean validatePhone(String s) {

        boolean b= false;
```

```
        if(s.length()==10)
        {
        if(s.matches("(0){2}[0-9]{8}"))
                b=false;
        else if(s.matches("[0-9]{10}"))
                b=true;
        else
                ;
}
        return b;
}
}
```

**39. String Splitter**

Write a program which would accept a string and a character as a delimiter. Apply the below rules

1. Using the delimiter, split the string and store these elements in array.
2. Reverse each element of the string and convert it into lowercase.

Include a class UserMainCode with a static method **manipulateLiteral** which accepts the string and character. The return type is the string array formed.

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

**Input and Output Format:**

Input consists of a string and character.
Output consists of a string array.

Refer sample output for formatting specifications.

**Sample Input 1:**

AAA/bba/ccc/DDD

/

**Sample Output 1:**

aaa

abb

ccc

ddd

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);

String s1=sc.next();
char s2=sc.next().charAt(0);
String res[]=User.manipulateLiteral (s1,s2);
for(int i=0;i<res.length;i++)
      System.out.println(res[i]);

}
}


publicclass User {
publicstatic String[] manipulateLiteral(String s1,char s2) {
      String ss=Character.toString(s2);
      StringTokenizer st=new StringTokenizer(s1,ss);
      ArrayList<String> a=new ArrayList<String>();
      while(st.hasMoreTokens())
      {
            StringBuffer sb=new StringBuffer();
            sb.append(st.nextToken().toLowerCase());
            a.add(sb.reverse().toString());

      }
      String[] s=new String[a.size()];
      for(int i=0;i<a.size();i++)
       s[i]=(String)a.get(i);


return s;
}
}
```

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

publicclass User {
publicstatic String[] manipulateLiteral(String s1,char s2) {
	String ss=String.valueOf(s2);
	StringTokenizer st=new StringTokenizer(s1,ss);
	ArrayList<String>a=new ArrayList<String>();

	while(st.hasMoreTokens())
	{		StringBuffer sb=new StringBuffer();
		sb.append(st.nextToken());
		a.add(sb.reverse().toString().toLowerCase());

	}
	int d=a.size();
	System.out.println(d);
	String[] s=new String[d];
	for(int i=0;i<a.size();i++)
	{
	 s[i]=a.get(i);
	}


return s;
}
}
```

**40. Vowel Count**

Write a program to read a string and count the number of vowels present in it.

Include a class UserMainCode with a static method **tellVowelCount** which accepts the string. The return type is the integer giving out the count of vowels.

Note: The check is case-insensitive.

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:**

NewYork

**Sample Output 1:**

2

**Sample Input 2:**

Elephant

**Sample Output 2:**

3

```
import java.util.Scanner;

public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);

String s1=sc.next();

System.out.println(User.tellVowelCount(s1));
}
}

publicclass User {
publicstaticint tellVowelCount(String s1) {
	int count=0;
	String s="aeoiu";
	String ss="AEIOU";
	for(int i=0;i<s1.length();i++)
	{
		for(int j=0;j<s.length();j++)
		{
			if(s1.charAt(i)==s.charAt(j) || s1.charAt(i)==ss.charAt(j)
)
				count++;
		}
	}
	return count;
}
}
```

```
publicstaticint tellVowelCount(String s1) {
      int count=0;
      for(int i=0;i<s1.length();i++)
      {
                  if(s1.charAt(i)=='a' || s1.charAt(i)=='e' ||
s1.charAt(i)=='i'
                        || s1.charAt(i)=='o' || s1.charAt(i)=='u' ||
                        s1.charAt(i)=='A' || s1.charAt(i)=='E' ||
                        s1.charAt(i)=='I' ||
                        s1.charAt(i)=='O' || s1.charAt(i)=='U' )

                  {
                        count++;
                  }
            }

      return count;
}
}
```

**********