31. Shift Left

Write a program to read a integer array of scores, and return a version of the given array where all the 5's have been removed. The remaining elements should shift left towards the start of the array as needed,

and the empty spaces at the end of the array should be filled with 0.

So {1, 5, 5, 2} yields {1, 2, 0, 0}.

Include a class UserMainCode with a static method shiftLeft which accepts the integer array. The return type is modified array.

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 modified array.
Refer sample output for formatting specifications.

Sample Input 1:
7
1
5
2
4
5
3
5

Sample Output 1:
1
2
4
3
0
0
0

Solution:

```
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 res[]=User.shiftLeft(a,n);
		for(int i=0;i<res.length;i++)
			System.out.println(res[i]);

	}

}



public class User {
	public static int[] shiftLeft(int a[],int n)
	{
		int b[]=new int[n];
		int k=0;
		for(int i=0;i<n;i++)
		{
			if(a[i]!=5)
			{
			b[k]=a[i];
			k++;
		}
		}
		return b;
	}

}
```

32. Word Count

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

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
String[] s = new String[n];
for (int i = 0; i < n; i++)
s[i] = sc.next();
System.out.println(User.sumOfDigits(s));

}
}

publicclass User {
```

```
publicstaticint sumOfDigits(String[] ss) {
int sum = 0, n = 0;
for (int i = 0; i < ss.length; i++) {
String s = ss[i];
for (int k = 0; k < s.length(); k++) {
if (Character.isDigit(s.charAt(k))) {
n = Character.getNumericValue(s.charAt(k));
sum = sum + n;
}
}
}
return sum;
}
}
```

33. Prefix finder

Given a string array (s) with each element in the array containing 0s and 1s. Write a program to get the number of strings in the array where one String is getting as prefixed in other String in that array .
Example 1: Input: {10,101010,10001,1111} Output =2 (Since 10 is a prefix of 101010 and 10001)
Example 2: Input: {010,1010,01,0111,10,10} Output =3(01 is a prefix of 010 and 0111. Also, 10 is a prefix of 1010) Note: 10 is NOT a prefix for 10.

Include a class UserMainCode with a static method **findPrefix** 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 followed by the array.
Output consists of a integer .
Refer sample output for formatting specifications.

Sample Input 1:
4
0
1
11
110

Sample Output 1:
3

Solution:

```
import java.util.HashSet;
import java.util.Scanner;


public class Piddi {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int size = sc.nextInt();
String input[] = new String[size];
for (int i = 0; i < size; i++) {
input[i] = sc.next();
}



HashSet<String> hs = new HashSet<String>();
for (int i = 0; i < size; i++) {
hs.add(input[i]);
}


size = hs.size();
int i = 0;


int count = 0;
for (i = 0; i < size; i++) {
for (int j = 0; j < size; j++) {
if (input[i].equals(input[j]) == false) {
if (input[j].startsWith(input[i])) {
count++;
}
}
}
}
System.out.println(count);
}
}
```

34. Commons

Given two arrays of strings,return the count of strings which is common in both arrays. Duplicate entries are counted only once.
Include a class UserMainCode with a static method **countCommonStrings** which accepts the string arrays. The return type is the integer formed based on rules.
Create a Class Main which would be used to accept the string arrays 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 by the array.
Output consists of a integer .
Refer sample output for formatting specifications.

Sample Input 1:
3
a
c
e
3
b
d
e
Sample Output 1:
1

Sample Input 2:
5
ba
ba
black
sheep
wool
5
ba
ba
have
any
wool
Sample Output 2:
2

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
int n1 = sc.nextInt();
```

```
String[] s1 = new String[n1];
for (int i = 0; i < n1; i++) {
s1[i] = sc.next();
}
int n2 = sc.nextInt();
String[] s2 = new String[n2];
for (int i = 0; i < n2; i++) {
s2[i] = sc.next();
}
System.out.println(User.countCommonStrings(s1, s2, n1, n2));

}
}
```

```
import java.util.ArrayList;

publicclass User {
publicstaticint countCommonStrings(String[] a, String[] b, int n1, int n2) {
int count = 0;
ArrayList<String> al = new ArrayList<String>();
for (int i = 0; i < n1; i++) {
for (int j = 0; j < n2; j++) {
if (a[i].equalsIgnoreCase(b[j]))
if (!al.contains(b[i])) {
count++;
al.add(a[i]);
}
}
}
return count;
}
}
```

```
import java.util.HashSet;
import java.util.Iterator;
public class Palindrome {
            public static int removeDuplicate(String[] words1,String[]
words2)
            {

                                    int count=0;
                                  HashSet<String> s1=new HashSet<String>();
                                    HashSet<String> s2=new HashSet<String>();
                                    for(int i=0;i<words1.length;i++)
                                    {
```

```
                                s1.add(words1[i]);

                        }
                        for(int i=0;i<words2.length;i++)
                        {
                                s2.add(words2[i]);
                        }
                        Iterator<String> it1=s1.iterator();

                while(it1.hasNext())
                        {
                                String its1=it1.next();
                                Iterator<String> it2=s2.iterator();
                                while(it2.hasNext())
                                {
                                        String its2=it2.next();
                                        if(its1.equals(its2))
                                        {
                                                count++;
                                        }
                                        }
                        }
                return count;
                }
}
```

35. Sequence Sum

Write a program to read a non-negative integer n, and find sum of fibonanci series for n number..

Include a class UserMainCode with a static method **getFibonacciSum** which accepts the integer value. The return type is integer.

The fibonacci seqence is a famous bit of mathematics, and it happens to have a recursive definition.

The first two values in the sequnce are 0 and 1.

Each subsequent value is the sum of the previous two values, so the whole seqence is 0,1,1,2,3,5 and so on.

You will have to find the sum of the numbers of the Fibonaaci series for a given int n.

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:

5

Sample Output 1:

7

Solution:

```
importjava.util.Scanner;

publicclass Main {

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


publicclass User {
publicstaticint getFibonacciSum(int n) {
int a = 0, b = 1, c = 0, d = 1;
for (int i = 3; i <= n; i++) {
c = a + b;
a = b;
b = c;
d = d + c;
}
return d;
}
}
```

36. Email Validation

Write a program to read a string and validate the given email-id as input.
Validation Rules:
1. Ensure that there are atleast 5 characters between '@' and '.'
2. There should be only one '.' and one '@' symbol.
3. The '.' should be after the '@' symbol.
4. There must be atleast three characters before '@'.
5. The string after '.' should only be 'com'

Include a class UserMainCode with a static method **ValidateEmail** which accepts the string. The return type is TRUE / FALSE as per problem.
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:
test@gmail.com
Sample Output 1:
TRUE

Sample Input 2:
academy@xyz.com
Sample Output 2:
FALSE

Solution:

```
import java.util.Scanner;

class Main {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
String email = sc.next();
System.out.println(User.ValidateEmail(email));
}
}
```

```
public class User {
public static boolean ValidateEmail(String email) {
boolean b = false;
if (email.matches("[a-zA-Z0-9]{3,}(@)[a-zA-Z]{5,}(.)(com)"))
b = true;
return b;
}
}
```

37. Symmetric Difference

Write a program to read two integer array and calculate the symmetric difference of the two arrays. Finally Sort the array.
Symmetric difference is the difference of A Union B and A Intersection B ie. [(A U B) - (A ^ B)]
Union operation merges the two arrays and makes sure that common elements appear only once.
Intersection operation includes common elements from both the arrays.
Ex - A={12,24,7,36,14} and B={11,26,7,14}.
A U B ={ 7,11,12,14,24,26,36} and
A ^ B = {7,14}
Symmetric difference of A and B after sorting= [A U B] - [A ^ B] = {11,12,24,26,36}.
Include a class UserMainCode with a static method **getSymmetricDifference** which accepts the integer array. The return type is an integer array.
Create a Class Main which would be used to accept the two integer arrays 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. The same sequnce is followed for the next array.
Output consists of sorted symmetric difference array.
Refer sample output for formatting specifications.

Sample Input 1:
5
11
5
14
26
3
3
5
3
1

Sample Output 1:

1
11
14
26

Solution:

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

int n1=sc.nextInt();
int[] a=new int[n1];
for(int i=0;i<n1;i++)
a[i]=sc.nextInt();
int n2=sc.nextInt();
int[] b= new int[n2];
for(int i=0;i<n2;i++)
b[i]=sc.nextInt();
int[] res=User.display(a,b,n1,n2);
for(int i=0;i<res.length;i++)
System.out.println(res[i]);



}
}
```

```
public class User {
public static int[] display(int a[],int b[],int n1,int n2)
{
TreeSet<Integer> ts1=new TreeSet<Integer>();
TreeSet<Integer> ts2=new TreeSet<Integer>();
TreeSet<Integer> ts3=new TreeSet<Integer>();
ArrayList<Integer> aa=new ArrayList<Integer>();
for(int i=0;i<a.length;i++)
ts1.add(a[i]);
for(int i=0;i<b.length;i++)
ts2.add(b[i]);
ts1.addAll(ts2);
for(int i=0;i<n1;i++)
{
for(int j=0;j<n2;j++)
```

```
{
if(a[i]==b[j])
ts3.add(a[i]);
}
}
ts1.removeAll(ts3);
aa.addAll(ts1);
int res[]=new int[aa.size()];
for(int i=0;i<res.length;i++)
res[i]=aa.get(i);
return res;
}
}
```

38. Day of Week

Write a program to read a string containing date in DD/MM/YYYY format and prints the day of the week that date falls on.
Return the day in lowercase letter (Ex: monday)

Include a class UserMainCode with a static method **getDayOfWeek** which accepts the string. The return type is the string.
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 a string.
Refer sample output for formatting specifications.

Sample Input 1:
02/04/1985
Sample Output 1:
tuesday

Solution:

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

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

```
            String s1=s.next();
            System.out.println(User.findOldDate(s1));
      }

}
```

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

public class User {
      public static String findOldDate(String s1) throws ParseException
      {
            SimpleDateFormat sd1=new SimpleDateFormat("dd-MM-yyyy");
            Date d1=sd1.parse(s1);
            SimpleDateFormat sd2=new SimpleDateFormat("EEEEE");
            String name=sd2.format(d1);
            return name.toLowerCase();
      }
}
```

39. Addtime

Write a program to read two String variables containing time intervals in hh:mm:ss format. Add the two time intervals and return a string in days:hours:minutes:seconds format where DD is number of days.
Hint: Maximum value for hh:mm:ss is 23:59:59

Include a class UserMainCode with a static method **addTime** which accepts the string values. The return type is the string.
Create a Class Main which would be used to accept the two string values and call the static method present in UserMainCode.

Input and Output Format:
Input consists of two string.
Output consists of a string.
Refer sample output for formatting specifications.

Sample Input 1:
12:45:30
13:50:45
Sample Output 1:
1:2:36:15

Sample Input 2:
23:59:59
23:59:59
Sample Output 2:
1:23:59:58

Solution:

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

publicclass Main {

publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = newScanner(System.in);
String s1 = sc.next();
String s2 = sc.next();
System.out.println(User.addTime(s1, s2));
}
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.TimeZone;

publicclass User {
publicstatic String addTime(String s1, String s2) throws ParseException {
// adding two times
SimpleDateFormat sdf = newSimpleDateFormat("HH:mm:ss");
sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
Date d1 = sdf.parse(s1);
Date d2 = sdf.parse(s2);
long time = d1.getTime() + d2.getTime();
String s = sdf.format(new Date(time));
// to calculate the day
Calendar c = Calendar.getInstance();
c.setTime(sdf.parse(s));
int day = c.get(Calendar.DAY_OF_MONTH);
if (day > 1)
day = day - 1;
String op = day + ":" + s;
return op;
}
```

}

```
import java.util.StringTokenizer;


public class Palindrome {
            public static String removeDuplicate(String a,String b)
            {

                  StringTokenizer st1=new StringTokenizer(a,":");
                  StringTokenizer st2=new StringTokenizer(b,":");

                  int h1=Integer.parseInt(st1.nextToken());
                  int m1=Integer.parseInt(st1.nextToken());
                  int s1=Integer.parseInt(st1.nextToken());
                  int d=0;
                  int h2=Integer.parseInt(st2.nextToken());
                  int m2=Integer.parseInt(st2.nextToken());
                  int s2=Integer.parseInt(st2.nextToken());

                  int m,h,s;
                  m=m1+m2;
                  h=h1+h2;
                  s=s1+s2;

                  if(s>=60)
                  {
                        m=m+1;
                        s=s-60;
                        if(m1>=60)
                        {
                              h=h+1;
                              m=m-60;
                              if(h>=24)
                              {
                                    d=d+1;
                                    h=h-24;
                              }
                        }
                  }

                  if(m1>=60)
                  {
                        h=h+1;
                        m=m-60;
                        if(h>=24)
                        {
                              d=d+1;
                              h=h-24;
```

```
                        }
                }

                if(h>=24)
                {
                        d=d+1;
                        h=h-24;
                }

                StringBuffer sb=new StringBuffer();

      sb.append(d).append(":").append(h).append(":").append(m).append(":").ap
pend(s);

                return sb.toString();


                }
}
```

```
0:00:01
0:00:02
0:0:0:3
```

```
12:45:30
13:50:45
1:2:36:15
```

12:20:20

22:20:10

```
12:20:20
22:20:10
1:10:40:30
```

1:20:20

2:20:10

```
1:20:20
2:20:10
0:3:40:30
```

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

```
public class Palindrome {
            public static String removeDuplicate(String a,String b)
            {
                  StringTokenizer st1=new StringTokenizer(a,":");
                  StringTokenizer st2=new StringTokenizer(b,":");

                  int h1=Integer.parseInt(st1.nextToken());
                  int m1=Integer.parseInt(st1.nextToken());
                  int s1=Integer.parseInt(st1.nextToken());
                  int d=0;
                  int h2=Integer.parseInt(st2.nextToken());
                  int m2=Integer.parseInt(st2.nextToken());
                  int s2=Integer.parseInt(st2.nextToken());

                  int m,h,s;
                  m=m1+m2;
                  h=h1+h2;
                  s=s1+s2;

                  while(s>=60)
                  {
                        m=m+1;
                        s=s-60;
                  }
                  while(m>=60)
                        {
                              h=h+1;
                              m=m-60;
                        }

                  while(h>=24)
                        {
                              d=d+1;
                              h=h-24;
                        }

                  StringBuffer sb=new StringBuffer();

      sb.append(d).append(":").append(h).append(":").append(m).append(":").ap
pend(s);

                  return sb.toString();


                  }
}
```

40. ISBN Validation

Write a program to read a string and validate the given ISBN as input.
Validation Rules:
1. An ISBN (International Standard Book Number) is a ten digit code which uniquely identifies a book.
2. To verify an ISBN you calculate 10 times the first digit, plus 9 times the second digit, plus 8 times the third ..all the way until you add 1 times the last digit.
If the final number leaves no remainder when divided by 11 the code is a valid ISBN.
Example 1:
Input:0201103311
Calculation: 10*0 + 9*2 + 8*0 + 7*1 + 6*1 + 5*0 + 4*3 + 3*3 + 2*1 + 1*1= 55.
55 mod 11 = 0
Hence the input is a valid ISBN number
Output: true
Include a class UserMainCode with a static method **validateISBN** which accepts the string. The return type is TRUE / FALSE as per problem.
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:
0201103311
Sample Output 1:
TRUE

Solution:

```
import java.util.Scanner;

public class Main {
	public static void main(String[] args) {
		Scanner s=new Scanner(System.in);
		String isbn=s.next();
		boolean b=User.validateISBN(isbn);
		System.out.println(b);
	}

}


import java.util.*;
```

```java
public class User {
	public static boolean validateISBN(String isbn)
	{
		int sum=0,k=10;
		for(int i=0;i<isbn.length();i++)
		{
			int a=Character.getNumericValue(isbn.charAt(i));
			sum=sum+(a*k);
			k--;
		}
		if(sum%11==0)
			return true;
		else
			return false;
	}
}
```