1. **Start Case**

Write a program to read a sentence in string variable and convert the first letter of each word to capital case. Print the final string.

Note: - Only the first letter in each word should be in capital case in final string.

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

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

**Input and Output Format:**

Input consists of a strings.

Output consists of a String (capitalized string).

Refer sample output for formatting specifications.

**Sample Input:**

Now is the time to act!

**Sample Output:**

Now Is The Time To Act!

**Solution :**

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

String s = sc.nextLine();

System.*out*.println(User.*printCapitalized*(s));

}

}

**import** java.util.StringTokenizer;

**publicclass** User {

**publicstatic** String printCapitalized(String s) {

StringTokenizer st= **new** StringTokenizer(s," ");

StringBuffer sb= **new** StringBuffer();

**while**(st.hasMoreTokens())

{

String s1=st.nextToken();

String s2=s1.substring(0, 1);

String s3=s1.substring(1);

sb.append(s2.toUpperCase());

sb.append(s3);

sb.append(" ");

}

**return** sb.toString();

}

}

**import** java.util.StringTokenizer;

**publicclass** UserMainCode {

**publicstatic** String changeWord(String s)

{

StringTokenizer st=**new** StringTokenizer(s," ");

StringBuffer sb=**new** StringBuffer();

**while**(st.hasMoreTokens())

{

String s1=st.nextToken();

sb.append(s1.substring(0,1).toUpperCase());

sb.append(s1.substring(1));

sb.append(" ");

}

**return** sb.toString();

}

}

1. **Maximum Difference**

Write a program to read an integer array and find the index of larger number of the two adjacent numbers with largest difference. Print the index.

Include a class **UserMainCode** with a static method **findMaxDistance**which accepts an integer array and the number of elements in the array. The return type (Integer) should return index.

Create a Class Main which would be used to accept an integer array and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of n+1 integers, where n corresponds the size of the array followed by n integers.

Output consists of an Integer (index).

Refer sample output for formatting specifications.

**Sample Input :**

6

4

8

6

1

9

4

**Sample Output :**

4

[In the sequence 4 8 6 1 9 4 the maximum distance is 8 (between 1 and 9). The function should return the index of the greatest of two. In this case it is 9 (which is at index 4). output = 4.]

**Solution :**

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

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

**int**[] a= **newint**[n];

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

a[i]=sc.nextInt();

System.*out*.println(User.*findMaxDistance*(a));

}

}

**publicclass** User {

**publicstaticint** findMaxDistance(**int**[] a){

**int** dif,max=0;

**int** n=a.length;

**for**(**int** i=0;i<n-1;i++)

{

dif=Math.*abs*(a[i]-a[i+1]);

// **if**(max<dif)

**if**(dif>max)

{

if(a[i+1]>a[i])

max=i+1;

else

max=i;

}

}

**return** max;

}

}

1. **Palindrome - In Range**

Write a program to input two integers, which corresponds to the lower limit and upper limit respectively, and find the sum of all palindrome numbers present in the range including the two numbers. Print the sum.

Include a class **UserMainCode** with a static method **addPalindromes**which accepts two integers. The return type (Integer) should return the sum if the palindromes are present, else return 0.

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

Note1 : A palindrome number is a number which remains same after reversing its digits.

Note2 : A single digit number is not considered as palindrome.

**Input and Output Format:**

Input consists of 2 integers, which corresponds to the lower limit and upper limit respectively.

Output consists of an Integer (sum of palindromes).

Refer sample output for formatting specifications.

**Sample Input :**

130

150

**Sample Output :**

272

**(131+141 = 272)**

**Solution:**

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

**int** a = sc.nextInt();

**int**b = sc.nextInt();

System.*out*.println(User.*addPalindromes*(a,b));

}

}

**publicclass** User {

**publicstaticint** addPalindromes(**int** a, **int** b){

**int** temp=0,sum=0,r,sum1=0;

**for**(**int** i=a;i<=b;i++)

{

temp=i;

sum=0;

**while**(temp>0){

r=temp%10;

sum=sum\*10+r;

temp=temp/10;

}

**if**(i==sum)

sum1=sum1+i;

}

**return** sum1;

}

}

1. **PAN Card**

Write a program to read a string and validate PAN no. against following rules:

1. There must be eight characters.

2. First three letters must be alphabets followed by four digit number and ends with alphabet

3. All alphabets should be in capital case.

Print “Valid” if the PAN no. is valid, else print “Invalid”.

Include a class **UserMainCode** with a static method **validatePAN**which accepts a string. The return type (Integer) should return 1 if the string is a valid PAN no. else return 2.

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

**Input and Output Format:**

Input consists of a string, which corresponds to the PAN number.

Output consists of a string - "Valid" or "Invalid"

Refer sample output for formatting specifications.

**Sample Input 1:**

ALD3245E

**Sample Output 1:**

Valid

**Sample Input 2:**

OLE124F

**Sample Output 2:**

Invalid

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

String s=sc.next();

**int** res=User.*validatePAN*(s);

**if**(res==1)

System.*out*.println("Valid");

**else**

System.*out*.println("Invalid");

}

}

**publicclass** User {

**publicstaticint** validatePAN(String s){

**int** res=0;

**if**(s.length()==8)

{

**if**(s.matches("[A-Z]{3}[0-9]{4}[A-Z]{1}"))

res=1;

**else**

res=2;

}

**return** res;

}

}

1. **Fibonacci Sum**

Write a program to read an integer n, generate fibonacci series and calculate the sum of first n numbers in the series. Print the sum.

Include a class **UserMainCode** with a static method **getSumOfNfibos**which accepts an integer n. The return type (Integer) should return the sum of n fibonacci numbers.

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

**Note:**First two numbers in a Fibonacci series are 0, 1 and all other subsequent numbers are sum of its previous two numbers. Example - 0, 1, 1, 2, 3, 5...

**Input and Output Format:**

Input consists of an integer, which corresponds to n.

Output consists of an Integer (sum of fibonacci numbers).

Refer sample output for formatting specifications.

**Sample Input :**

5

**Sample Output :**

7

**[0 + 1 + 1 + 2 + 3 = 7]**

Solutions:

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

**int** s=sc.nextInt();

System.*out*.println(User.*getSumOfNfibos*(s));

}

}

**publicclass** User {

**publicstaticint** getSumOfNfibos(**int** s){

**int** a=0,b=1,c=0,d=1;

**for**(**int** i=3;i<=s;i++)

{

c=a+b;

a=b;

b=c;

d=d+c;

}

**return**d;

}

}

1. **Test Vowels**

Write a program to read a string and check if given string contains exactly five vowels in any order. Print “Yes” if the condition satisfies, else print “No”.

Assume there is no repetition of any vowel in the given string and all characters are lowercase.

Include a class **UserMainCode** with a static method **testVowels**which accepts a string. The return type (Integer) should return 1 if all vowels are present, else return 2.

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

**Input and Output Format:**

Input consists of a string.

Output consists of a string (“Yes” or “No”).

Refer sample output for formatting specifications.

**Sample Input 1:**

acbisouzze

**Sample Output 1:**

Yes

**Sample Input 2:**

cbisouzze

**Sample Output 2:**

No

Solutions:

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

String s=sc.next();

**int** res=User.*testVowels* (s);

**if**(res==1)

System.*out*.println("Yes");

**else**

System.*out*.println("No");

}

}

**publicclass** User {

**publicstaticint** testVowels (String s){

**int** res,count=0;

String s1="aeiou";

String s2=s.toLowerCase();

**for**(**int** i=0;i<s2.length();i++)

{

**for**(**int** j=0;j<s1.length();j++)

{

**if**(s2.charAt(i)==s1.charAt(j))

{

count++;

}

}

}

**if**(count==s1.length())

res=1;

**else**

res=2;

**return** res;

}

}

**publicclass** User {

**publicstaticint** testOrderVowels(String s1) {

StringBuffer sb = **new**StringBuffer();

**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;

}

}

**7 . Dash Check**

Write a program to read two strings and check whether or not they have dashes in the same places. Print “Yes” if the condition satisfies, else print “No”.

Include a class **UserMainCode** with a static method **compareDashes**which accepts two strings. The return type (Integer) should return 1 if all dashes are placed correctly, else return 2.

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

**Note:**The strings must have exactly the same number of dashes in exactly the same positions. The strings might be of different length.

**Input and Output Format:**

Input consists of two strings.

Output consists of a string (“Yes” or “No”).

Refer sample output for formatting specifications.

**Sample Input 1:**

hi—there-you.

12--(134)-7539

**Sample Output 1:**

Yes

**Sample Input 2:**

-15-389

-xyw-zzy

**Sample Output 2:**

No

Solution:

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

String s1=sc.next();

String s2=sc.next();

**int** res=User.compareDashes (s1,s2);

**if**(res==1)

System.*out*.println("Yes");

**else**

System.*out*.println("No");

}

}

**import** java.util.ArrayList;

**publicclass** User {

**publicstaticint** compareDashes (String s1,String s2){

**int** res=0;

ArrayList<Integer> a1=**new** ArrayList<Integer>();

ArrayList<Integer> a2=**new** ArrayList<Integer>();

**for**(**int** i=0;i<s1.length();i++)

{

**if**(s1.charAt(i)=='-')

a1.add(i);

}

**for**(**int** i=0;i<s2.length();i++)

{

**if**(s2.charAt(i)=='-')

a2.add(i);

}

**if**(a1.equals(a2))

res=1;

**else**

res=2;

**return** res;

}

}

1. **Reverse Split**

Write a program to read a string and a character, and reverse the string and convert it in a format such that each character is separated by the given character. Print the final string.

Include a class **UserMainCode** with a static method **reshape**which accepts a string and a character. The return type (String) should return the final string.

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

**Input and Output Format:**

Input consists of a string and a character.

Output consists of a string (the final string).

Refer sample output for formatting specifications.

**Sample Input:**

Rabbit

-

**Sample Output:**

t-i-b-b-a-R

**import**java.util.HashMap;

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

String s1=sc.next();

String s2=sc.next();

System.*out*.println(User.*extractMax*(s1,s2));

}

}

**publicclass** UserMain {

**publicstatic**String extractMax(String s1,String s2){

StringBuffer sb=**new** StringBuffer();

**for**(**int** i=0;i<s1.length()-1;i++)

{

sb.append(s1.charAt(i));

sb.append(s2);

}

sb.append(s1.charAt(s1.length()-1));

**return** sb.reverse().toString();

}

}

1. **Remove 10's**

Write a program to read an integer array and remove all 10s from the array, shift the other elements towards left and fill the trailing empty positions by 0 so that the modified array is of the same length of the given array.

Include a class **UserMainCode** with a static method **removeTens**which accepts the number of elements and an integer array. The return type (Integer array) should return the final array.

Create a Class Main which would be used to read the number of elements and the input array, and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of n+1 integers, where n corresponds to size of the array followed by n elements of the array.

Output consists of an integer array (the final array).

Refer sample output for formatting specifications.

**Sample Input :**

5

1

10

20

10

2

**Sample Output :**

1

20

2

o

o

Solution :

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

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

**int**[] a= **newint**[n];

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

a[i]=sc.nextInt();

User.*removeTens*(a);

}

}

**import**java.util.ArrayList;

**publicclass** User {

**publicstaticint**[] removeTens(**int**[] a){

**int**[] out=**newint**[a.length];

**int** k=0;

**for**(**int** i=0;i<a.length;i++)

{

**if**(a[i]!=10)

{

out[k]=a[i];

k++;

}}

**for**(**int** i=0;i<a.length;i++)

System.*out*.println(out[i]);

**return** out;

}

}

1. **Last Letters**

Write a program to read a sentence as a string and store only the last letter of each word of the sentence in capital letters separated by $. Print the final string.

Include a class **UserMainCode** with a static method **getLastLetter**which accepts a string. The return type (string) should return the final string.

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

**Input and Output Format:**

Input consists of a string.

Output consists of a string (the final string).

Refer sample output for formatting specifications.

**Smaple Input :**

This is a cat

**Sample Output :**

S$S$A$T

Solution :

**import**java.util.ArrayList;

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

String s= sc.nextLine();

System.*out*.println(User.*getLastLetter*(s));

}

}

**import** java.util.StringTokenizer;

**publicclass** User {

**publicstatic** String getLastLetter(String s){

StringTokenizer st= **new** StringTokenizer(s," ");

String s2=st.nextToken();

StringBuffer sb= **new** StringBuffer();

String s3=Character.*toUpperCase*(s2.charAt(s2.length()-1))+"";

**while**(st.hasMoreTokens())

{

s2=st.nextToken();

s3=s3+"$"+Character.*toUpperCase*(s2.charAt(s2.length()-1));

}

**return**s3;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**publicclass** UserMain {

**publicstatic** String getLastLetter(String s)

{

StringTokenizer st= **new** StringTokenizer(s," ");

StringBuffer sb= **new** StringBuffer();

String b=st.nextToken();

sb.append(b.charAt(b.length()-1));

**while**(st.hasMoreTokens())

{

String a=st.nextToken();

sb.append("$");

sb.append(a.charAt(a.length()-1));

}

**return** sb.toString().toUpperCase();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**11 Largest Key in HashMap**

Write a program that construts a hashmap and returns the value corresponding to the largest key.

Include a class UserMainCode with a static method **getMaxKeyValue** which accepts a string. The return type (String) should be the value corresponding to the largest key.

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 2n+1 values. The first value corresponds to size of the hashmap. The next n pair of numbers equals the integer key and value as string.

Output consists of a string which is the value of largest key.

Refer sample output for formatting specifications.

**Sample Input 1:**

3

12

amron

9

Exide

7

SF

**Sample Output 1:**

amron

Solutions:

import java.util.HashMap;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n= sc.nextInt();

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

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

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

System.out.println(User.getMaxKeyValue(hm));

}

}

**import**java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import**java.util.StringTokenizer;

**publicclass** User {

**publicstatic**String getMaxKeyValue(HashMap<Integer,String> hm){

**int** max=0;

String nn=**null**;

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

**while**(it.hasNext())

{

**int** key=it.next();

String name=hm.get(key);

**if**(key>max)

{

key=max;

nn=name;

}

}

**return** nn;

}

}

**12. All Numbers**

Write a program to read a string array and return 1 if all the elements of the array are numbers, else return -1.

Include a class UserMainCode with a static method **validateNumber** which accepts a string aray. The return type (integer) should be -1 or 1 based on the above rules.

Create a Class Main which would be used to accept Input string array and call the static method present in UserMainCode.  
The string array is said to be valid if all the elements in the array are numbers. Else it is invalid.

**Input and Output Format:**

Input consists of an integer specifying the size of string array followed by n strings.

Refer sample output for formatting specifications.

**Sample Input 1:**

4  
123

24.5

23

one

**Sample Output 1:**

invalid

**Sample Input 2:**

2  
123

24.5

**Sample Output 2:**

valid

import java.util.HashMap;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n= sc.nextInt();

String[] s= new String[n];

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

s[i]=sc.next();

int res=User.validateNumber(s);

if(res==1)

System.out.println("Valid");

else

System.out.println("invalid");

}

}

**import**java.util.ArrayList;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.StringTokenizer;

**publicclass** User {

**publicstaticint** validateNumber(String s[]){

**int** res=0;

**int** count=0,temp=0;

String s1=**null**;

**for**(**int** i=0;i<s.length;i++)

{

s1=s[i];

count=0;

**for**(**int** j=0;j<s1.length();j++)

{

**if**(s1.charAt(j)>='0'&& s1.charAt(j)<='9' || s1.charAt(j)=='.')

count++;

}

**if**(count==s1.length())

temp++;

}

**if**(temp==s.length)

res=1;

**else**

res=-1;

**return** res;

}

}

**13. Day of the Week**

Write a program to read a date as string (MM-dd-yyyy) and return the day of week on that date.

Include a class UserMainCode with a static method **getDay** which accepts the string. The return type (string) should be the day of the week.

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 1:**

07-13-2012

**Sample Output 1:**

Friday

 Solutions :

User :

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

public class User

{

public static String calculateBornDay(String d) throws ParseException

{

SimpleDateFormat sdf= new SimpleDateFormat("MM-dd-yyyy");

SimpleDateFormat s= new SimpleDateFormat("EEEE");

Date d1= new Date();

d1= sdf.parse(d);

String day=s.format(d1);

return day;

}

}

**14. Max Substring**

Write a program to accept two string inputs. The first being a source string and second one a delimiter. The source string contains the delimiter at various locations. Your job is to return the substring with maximum number of characters. If two or more substrings have maximim number of characters return the substring which appears first. The size of the delimiter is 1.

Include a class UserMainCode with a static method **extractMax** which accepts the string. The return type (string) should be the max substring.

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 source string and delimiter.

Output consists of a string.

Refer sample output for formatting specifications.

**Sample Input 1:**

delhi-pune-patna

-

**Sample Output 1:**

Delhi

**import**java.util.HashMap;

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

String s1=sc.next();

String s2=sc.next();

System.*out*.println(User.*extractMax*(s1,s2));

}

}

**import**java.util.ArrayList;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import** java.util.StringTokenizer;

**publicclass** User {

**publicstatic** String extractMax(String s1,String s2){

StringTokenizer st= **new** StringTokenizer(s1,s2);

**int** max=0,c=0;

String str=**null**;

**while**(st.hasMoreTokens())

{

String s= st.nextToken();

c=s.length();

**if**(c>max)

{

c=max;

str=s;

}

}

**return** str;

}

}

**15. States and Capitals**

Write a program that construts a hashmap with “state” as key and “capital” as its value. If the next input is a state, then it should return capital$state in lowercase.

Include a class UserMainCode with a static method **getCapital** which accepts a hashmap. The return type is the string as given in the above statement

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 2n+2 values. The first value corresponds to size of the hashmap. The next n pair of numbers contains the state and capital. The last value consists of the “state” input.

Output consists of a string as mentioned in the problem statement.

Refer sample output for formatting specifications.

**Sample Input 1:**

3

Karnataka

Bangaluru

Punjab

Chandigarh

Gujarat

Gandhinagar

Punjab

**Sample Output 1:**

chandigarh$punjab

 **import** java.util.HashMap;

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

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

HashMap<String,String> hm= **new** HashMap<String,String>();

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

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

String s=sc.next();

System.*out*.println(User.*getCapital*(hm,s));

}

}

**import**java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import**java.util.StringTokenizer;

**publicclass** User {

**publicstatic** String getCapital(HashMap<String,String> hm,String s){

Iterator<String> it=hm.keySet().iterator();

StringBuffer sb= **new** StringBuffer();

**while**(it.hasNext())

{

String state=it.next();

String cap=hm.get(state);

**if**(state.equalsIgnoreCase(s))

{

sb.append(cap).append('$').append(state);

}

}

**return** sb.toString().toLowerCase();

}

}

**16. Simple String Manipulation - II**

Write a program to read a string and return an integer based on the following rules.

If the first word and the last word in the String match, then return the number of characters in the word else return sum of the characters in both words. Assume the Strings to be case - sensitive.

Include a class UserMainCode with a static method **calculateWordSum** which accepts a string. The return type (integer) should be based on the above rules.

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 with maximum size of 100 characters.

Output consists of a string.

Refer sample output for formatting specifications.

**Sample Input 1:**

COGNIZANT TECHNOLOGY SOLUTIONS COGNIZANT

**Sample Output 1:**

9

**Sample Input 2:**

HOW ARE YOU

**Sample Output 2:**

6

**import**java.util.HashMap;

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

String s=sc.nextLine();

System.*out*.println(User.*calculateWordSum* (s));

}

}

**import**java.util.ArrayList;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import** java.util.StringTokenizer;

**publicclass** User {

**publicstaticint** calculateWordSum (String s){

**int** sum=0,i=0;

StringTokenizer st=**new** StringTokenizer(s," ");

String[] s1= **new** String[st.countTokens()];

**while**(st.hasMoreTokens())

{

s1[i]=st.nextToken();

i++;

}

**if**(s1[0].equals(s1[s1.length-1]))

sum=s1[0].length();

**else**

sum=s1[0].length()+s1[s1.length-1].length();

**return** sum;

}

}

**17. Vowels, Arrays & ArrayLists**

Write a program to read an array of strings and return an arraylist which consists of words whose both first and last characters are vowels. Assume all inputs are in lowecase.

Include a class UserMainCode with a static method **matchCharacter** which accepts a string array. The return type shoud be an arraylist which should contain elements as mentioned above.

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' string correspond to the elements in the array.

Output consists of strings which are elements of arraylist

Refer sample output for formatting specifications.

**Sample Input 1:**

4

abcde

pqrs

abci

orto

**Sample Output 1:**

abcde

abci

orto

import java.util.HashMap;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(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.matchCharacter (s));

}

}

**import** java.util.ArrayList;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.StringTokenizer;

**publicclass** User {

**publicstatic** ArrayList<String> matchCharacter (String[] s){

ArrayList<String> a= **new** ArrayList<String>();

**for**(**int** i=0;i<s.length;i++)

{

System.*out*.println(s[i].charAt(0));

System.*out*.println(s[i].charAt(s[i].length()-1));

**if**((s[i].charAt(0)=='a'|| s[i].charAt(0)=='e'||

s[i].charAt(0)=='i'||s[i].charAt(0)=='o'||

s[i].charAt(0)=='u') && (s[i].charAt(s[i].length()-1)=='a'||

s[i].charAt(s[i].length()-1)=='e'||s[i].charAt(s[i].length()-1)=='i'||

s[i].charAt(s[i].length()-1)=='o'||s[i].charAt(s[i].length()-1)=='u'))

{

a.add(s[i]);

}

}

**return** a;

}

}

**18.Transfer from Hashmap to Arraylist**

Write a program that constructs a hashmap with “employee id” as key and “name” as its value. Based on the rules below, on being satisfied, the name must be added to the arraylist.

i)First character should be small and the last character should be Capital.

ii)In name at least one digit should be there.

Include a class UserMainCode with a static method **getName** which accepts a hashmap. The return type is an arraylist as expected in the above statement

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 2n+1 values. The first value corresponds to size of the hashmap. The next n pair of numbers contains the employee id and name.

Output consists of arraylist of strings as mentioned in the problem statement.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

1

ravi5raJ

2

sita8gitA

3

ram8sitA

4

rahul

**Sample Output 1:**

ravi5raJ

sita8gitA

ram8sitA

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n=sc.nextInt();

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

ArrayList<String> a= new ArrayList<String>();

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

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

a=User.getName(hm);

for(int i=0;i<a.size();i++)

{

System.out.println(a.get(i));

}

}

}

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import**java.util.StringTokenizer;

**publicclass** User {

**publicstatic** ArrayList<String> getName(HashMap<Integer,String> hm){

ArrayList<String> a= **new** ArrayList<String>();

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

**while**(it.hasNext())

{

**int** id=it.next();

String name=hm.get(id);

**for**(**int** i=0;i<name.length();i++)

{

**if**(name.charAt(0)>=97 && name.charAt(0)<=122 &&

name.charAt(name.length()-1)>=65 && name.charAt(name.length()-1)<=96)

{

**if**(name.charAt(i)>='0'&& name.charAt(i)<='9')

{

a.add(name);

}

}

}

}

**return** a;

}

}

**19. Max Admissions**

Write a program that reads details about number of admissions per year of a particular college, return the year which had maximum admissions. The details are stored in an arraylist with the first index being year and next being admissions count.

Include a class UserMainCode with a static method **getYear** which accepts a arraylist. The return type is an integer indicating the year of max admissions.

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 2n+1 values. The first value corresponds to size of the data (year & admissions). The next n pair of numbers contains the year and admissions count.

Output consists of an integer as mentioned in the problem statement.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

2010

200000

2011

300000

2012

45000

2013

25000

**Sample Output 1:**

2011

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n=sc.nextInt();

ArrayList<Integer> a= new ArrayList<Integer>();

for(int i=0;i<n\*2;i++)

a.add(sc.nextInt());

System.out.println(User.getYear(a));

}

}

**import** java.util.ArrayList;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.StringTokenizer;

**publicclass** User {

**publicstaticint** getYear(ArrayList<Integer> a){

**int**year=0;

**int** max=0;

**for**(**int** i=1;i<a.size();i=i+2)

{

**int** x=a.get(i);

**if**(x>max)

{

max=x;

year=a.get(i-1);

}

}

**return**year;

}

}

**20. Sum Non Prime Numbers**

Write a program to calculate the sum of all the non prime positive numbers less than or equal to the given number.

Note: prime is a natural number greater than 1 that has no positive divisors other than 1 and itself

Example:

input = 9

Prime numbers = 2,3,5 and 7

output = 1+4+6+8+9=28

Include a class **UserMainCode** with a static method “**addNumbers**” that accepts an integer arguement and returns an integer.

Create a class **Main** which would get an integer as input and call the static method **validateNumber** present in the UserMainCode.

**Input and Output Format:**

Input consists of an integer.

Output consists of an integer.

**Sample Input:**

9

**Sample Output:**

28

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n=sc.nextInt();

System.out.println(User.addNumbers(n));

}

}

**import**java.util.ArrayList;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.StringTokenizer;

**publicclass** User {

**publicstaticint** addNumbers(**int** n){

**int** c=0,sum=0;

**for** (**int** i = 1; i <= n; i++)

{

c=0;

**for**(**int** j=1;j<=i;j++)

{

**if**(i%j==0)

c++;

}

**if** (c==2)

;

**else**

sum=sum+i;

}

**return** sum;

}

}

--------------------------------------------

**21. Date Format Conversion**

Given a date string in the format dd/mm/yyyy, write a program to convert the given date to the format dd-mm-yy.

Include a class **UserMainCode** with a static method “**convertDateFormat**” that accepts a String and returns a String.

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

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input:**

12/11/1998

**Sample Output:**

12-11-98

import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class Main {

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

Scanner sc = new Scanner(System.in);

String n=sc.next();

System.out.println(User.convertDateFormat(n));

}

}

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import**java.util.ArrayList;

**import** java.util.Date;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.StringTokenizer;

**publicclass** User {

**publicstatic** String convertDateFormat(String n) **throws** ParseException{

SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");

Date d= sdf.parse(n);

SimpleDateFormat sdf1=**new** SimpleDateFormat("dd-MM-yyyy");

String s=sdf1.format(d);

**return**s;

}

}

**22. Valid Date**

Given a date string as input, write a program to validate if the given date is in any of the following formats:

dd.mm.yyyy

dd/mm/yy

dd-mm-yyyy

Include a class **UserMainCode** with a static method “**validateDate**” that accepts a String and returns an integer. This method returns 1 if the date is valid, else return -1.

Create a class **Main** which would get a String as input and call the static method **validateDate** 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:**

12.03.2012

**Sample Output 1:**

Valid

**Sample Input 2:**

27#01#1977

**Sample Output 2:**

Invalid

**publicstaticvoid** main(String[] args) {

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

String s=sc.nextLine();

SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");

sdf.setLenient(**false**);

**int** res=0;

**if**(s.matches("[0-9]{2}(/)[0-9]{2}(/)[0-9]{4}"))

{

**try** {

Date d=sdf.parse(s);

res=1;

} **catch** (ParseException e) {

res=-1;

}

System.*out*.println(res);

}

}

}

**23. Convert Format**

Given a 10 digit positive number in the format XXX-XXX-XXXX as a string input, write a program to convert this number to the format XX-XX-XXX-XXX.

Include a class **UserMainCode** with a static method “**convertFormat**” that accepts a String argument and returns a String.

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

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input:**

555-666-1234

**Sample Output:**

55-56-661-234

import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class Main {

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

Scanner sc = new Scanner(System.in);

String n=sc.next();

System.out.println(User.convertFormat(n));

}

}

**import** java.text.ParseException;

**import**java.text.SimpleDateFormat;

**import**java.util.ArrayList;

**import**java.util.Date;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import** java.util.StringTokenizer;

**publicclass** User {

**publicstatic** String convertFormat(String s) **throws** ParseException{

StringTokenizer st=**new** StringTokenizer(s,"-");

**int** i=0;

String[] s1=**new** String[st.countTokens()];

**while**(st.hasMoreTokens())

{

s1[i]=st.nextToken();

i++;

}

StringBuffer sb=**new** StringBuffer();

sb.append(s1[0].substring(0,2));

sb.append("-");

sb.append(s1[0].substring(2)).append(s1[1].substring(0,1));

sb.append("-");

sb.append(s1[1].substring(1)).append(s1[2].substring(0,1));

sb.append("-");

sb.append(s1[2].substring(1));

**return** sb.toString();

}

}

**import** java.util.StringTokenizer;

**publicclass** UserMainCode {

**publicstatic** String convertFormat(String s)

{

StringBuffer sb=**new** StringBuffer();

StringTokenizer st=**new** StringTokenizer(s,"-");

String s1=st.nextToken();

String s2=st.nextToken();

String s3=st.nextToken();

sb.append(s1.substring(0,2));

sb.append("-");

sb.append(s1.substring(s1.length()-1));

sb.append(s2.substring(0,1));

sb.append("-");

sb.append(s2.substring(1));

sb.append(s3.substring(0,1));

sb.append("-");

sb.append(s3.substring(1));

**return** sb.toString();

}

}

**24. Add and Reverse**

Given an int array and a number as input, write a program to add all the elements in the array greater than the given number. Finally reverse the digits of the obtained sum and print it.

Include a class **UserMainCode** with a static method “**addAndReverse**” that accepts 2 arguments and returns an integer.The first argument corresponds to the integer array and the second argument corresponds to the number.

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

**Example:**

Input Array = {10,15,20,25,30,100}

Number = 15

sum = 20 + 25 + 30 + 100 = 175

output = 571

**Input and Output Format:**

The first line of the input consists of an integer that corresponds to the number of elements in the array.

The next n lines of the input consists of integers that correspond to the elements in the array.

The last line of the input consists of an integer that corresponds to the number.

Output consists of a single integer.

**Sample Input**

6

10

15

20

25

30

100

15

**Sample Output**

571

import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

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();

int x=sc.nextInt();

System.out.println(User.addAndReverse(a,x));

}

}

**import**java.text.ParseException;

**import**java.text.SimpleDateFormat;

**import**java.util.ArrayList;

**import**java.util.Date;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import**java.util.StringTokenizer;

**publicclass** User {

**publicstaticint** addAndReverse(**int**[] a,**int** x) {

**int** sum=0;

**int** rev=0,r=0;

**for**(**int** i=0;i<a.length;i++)

{

**if**(x<a[i])

sum=sum+a[i];

}

**while**(sum!=0)

{

r=sum%10;

rev=rev\*10+r;

sum=sum/10;

}

**return** rev;

}

}

**25. Next Year day**

Given a date string in dd/mm/yyyy format, write a program to calculate the day which falls on the same date next year. Print the output in small case.

The days are sunday, monday, tuesday, wednesday, thursday, friday and saturday.

Include a class **UserMainCode** with a static method “**nextYearDay**” that accepts a String and returns a String.

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

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input:**

13/07/2012

**Sample Output:**

saturday

import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class Main {

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

Scanner sc = new Scanner(System.in);

String n=sc.next();

System.out.println(User.nextYearDay(n));

}

}

**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;

**publicclass** User {

**publicstatic** String nextYearDay(String s) **throws** ParseException {

SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");

SimpleDateFormat sdf1=**new** SimpleDateFormat("EEEE");

Date d= sdf.parse(s);

Calendar c=Calendar.*getInstance*();

c.setTime(d);

c.add(Calendar.*YEAR*,1);

Date year=c.getTime();

String day=sdf1.format(year);

**return** day;

}

}

**26. Sum Squares of Digits**

Write a program that accepts a positive number as input and calculates the sum of squares of individual digits of the given number.

Include a class **UserMainCode** with a static method “**getSumOfSquaresOfDigits**” that accepts an integer argument and returns an integer.

Create a class **Main** which would get an integer as input and call the static method **getSumOfSquaresOfDigits** present in the UserMainCode.

**Input and Output Format:**

Input consists of an integer.

Output consists of an integer.

**Sample Input:**

321

**Sample Output:**

14

import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

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();

System.out.println(User.getSumOfSquaresOfDigits(n));

}

}

**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;

**publicclass** User {

**publicstaticint** getSumOfSquaresOfDigits(**int** n) {

**int** sum=0,r=0;

**while**(n!=0)

{

r=n%10;

sum=sum+(r\*r);

n=n/10;

}

**return** sum;

}

}

**27. Even and Odd Index Sum**

Write a program that accepts a positive number as input and calculates the sum of digits at even indexes (say evenSum) and sum of digits at odd indexes (say oddSum) in the given number. If both the sums are equal , print 'yes', else print no.

Example:

input = 23050

evenSum = 2 + 0 + 0 = 2

oddSum = 3 + 5 = 8

output = no

Include a class **UserMainCode** with a static method “**sumOfOddEvenPositioned**” that accepts an integer and returns an integer. The method returns 1 if the 2 sums are equal. Else the method returns -1.

Create a class **Main** which would get an integer as input and call the static method **sumOfOddEvenPositioned** present in the UserMainCode.

**Input and Output Format:**

Input consists of an integer.

Output consists of a string that is either “yes” or “no”.

**Sample Input 1:**

23050

**Sample Output 1:**

no

**Sample Input 2:**

231

**Sample Output 2:**

yes

import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

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 res=User.sumOfOddEvenPositioned(n);

if(res==1)

System.out.println("yes");

else

System.out.println("no");

}

}

**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;

**publicclass** User {

**publicstaticint** sumOfOddEvenPositioned(**int** n) {

**int** even=0,odd=0;

**int** res=0,r=0,m=0;

**int** j=0,count=0;

**int** n1=n;

**while**(n>0)

{

n=n/10;

count++;

}

System.*out*.println(count);

**int**[] a=**newint**[count];

**while**(n1!=0)

{

r=n1%10;

a[j]=r;

j++;

n1=n1/10;

}

**int**[] b=**newint**[j];

**for**(**int** k=j-1;k>=0;k--)

{

b[m]=a[k];

m++;

}

**for**(**int** i=0;i<m;i++)

{

System.*out*.println("a:"+b[i]);

**if**(i%2==0)

even=even+b[i];

**else**

odd=odd+b[i];

}

System.*out*.println(even);

System.*out*.println(odd);

**if**(even==odd)

res=1;

**else**

res=-1;

**return** res;

}

}

**28. Remove 3 Multiples**

Write a program that accepts an ArrayList of integers as input and removes every 3rd element and prints the final ArrayList.

Suppose the given arrayList contains 10 elements remove the 3rd, 6th and 9th elements.

Include a class **UserMainCode** with a static method “**removeMultiplesOfThree**” that accepts an ArrayList<Integer> as arguement and returns an ArrayList<Integer>.

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

**Input and Output Format:**

The first line of the input consists of an integer n, that corresponds to the number of elements to be added in the ArrayList.

The next n lines consist of integers that correspond to the elements in the ArrayList.

Output consists of an ArrayList of integers.

**Sample Input:**

6

3

1

11

19

17

19

**Sample Output**

3

1

19

17

**publicclass** Main {

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

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

ArrayList<Integer> a=**new** ArrayList<Integer>();

ArrayList<Integer> res=**new** ArrayList<Integer>();

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

a.add(sc.nextInt());

res=User.*removeMultiplesOfThree*(a);

**for**(**int** i=0;i<res.size();i++)

System.*out*.println(res.get(i));

}

}

**publicclass** User {

**publicstatic** ArrayList<Integer> removeMultiplesOfThree(ArrayList<Integer> a) {

ArrayList<Integer> b=**new** ArrayList<Integer>();

**for**(**int** i=0;i<a.size();i++)

{

**int** d=a.get(i);

**if**(d%3!=0)

{

b.add(a.get(i));

}

}

**return** b;

}

}

**29.String Occurances - II**

Obtain two strings S1,S2 from user as input. Your program should count the number of times S2 appears in S1.  
  
Return the count as output. Note - Consider case.  
  
Include a class UserMainCode with a static method **getSubstring** which accepts two string variables. The return type is the count.  
  
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 two strings with maximum size of 100 characters.  
  
Output consists of an integer.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
catcowcat  
cat  
  
**Sample Output 1:**  
2  
  
  
**Sample Input 2:**  
catcowcat  
CAT  
  
**Sample Output 2:**  
0

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 s1=sc.next();

String s2=sc.next();

System.out.println(User.getSubstring(s1, s2));

}

}

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 int getSubstring (String s1,String s2) {

int count=0;

int n=s1.length()-(s2.length()-1);

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

{

String s3=s1.substring(i,i+(s2.length()));

if(s2.equals(s3))

count++;

}

return count;

}

}

**publicclass** User {

**publicstaticint** getSubstring (String s1,String s2) {

**int** count=0;

**int** n=s1.length()-(s2.length()-1);

**int**s2l=s2.length();

System.*out*.println(n);

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

{

String s3=s1.substring(i,i+s2l);

**if**(s2.equals(s3))

count++;

}

**return** count;

}

}

**30. Programming Logic**

Write a Program that accepts three integer values (a,b,c) and returns their sum. However, if one of the values is 13 then it does not count towards the sum and the next number also does not count. So for example, if b is 13, then both b and c do not count.  
  
Include a class UserMainCode with a static method **getLuckySum** which accepts three integers. The return type is integer representing the sum.  
  
Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of three integers.  
  
Output consists of a single integer.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
1  
2  
3  
  
**Sample Output 1:**  
6  
  
  
**Sample Input 2:**  
1  
2  
13  
  
**Sample Output 2:**  
3  
  
  
**Sample Input 3:**  
13  
3  
8  
  
**Sample Output 3:**  
8

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 int getLuckySum (int a,int b, int c) {

int res=0;

if(a==13)

res=c;

else if(b==13)

res=a;

else if(c==13)

res=a+b;

else

res=a+b+c;

return res;

}

}

----------------------------------------------------------

**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;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**41. Playing with String - II**

Write a program to accept a string array as input, convert all the elements into lowercase and sort the string array. Display the sorted array.  
  
Include a class UserMainCode with a static method **sortArray** which accepts the string array. The return type is the string array formed based on requirement.  
  
Create a Class Main which would be used to accept the string array 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,  
Output consists of a string array.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
5  
AAA  
BB  
CCCC  
A  
ABCDE  
  
**Sample Output 1:**  
a  
aaa  
abcde  
bb  
cccc

**publicclass** Main {

**publicstaticvoid** 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.*sortArray*(a);

**for**(**int** i=0;i<res.length;i++)

System.*out*.println(res[i]);

}

}

**publicclass** User {

**publicstatic** String[] sortArray (String s[]) {

String[] a=**new** String[s.length];

**for**(**int** i=0;i<s.length;i++)

{

a[i]=s[i].toLowerCase();

}

Arrays.*sort*(a);

**return** a;

}

}

**42. Median Calculation**

Write a program to accept an int array as input, and calculate the median of the same.  
  
Median Calculation Procedure:

1. Sort the sequence of numbers.
2. The total number count is odd, Median will be the middle number.

    The total number count is even, Median will be the average of two middle numbers, After calculating the average, round the number to nearest integer.  
  
Include a class UserMainCode with a static method **calculateMedian** which accepts the int array. The return type is the integer which would be the median.  
  
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 a an integer which denotes the size of the array followed by the array of integers.  
Output consists of a integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
7  
1  
2  
1  
4  
7  
1  
2  
  
**Sample Output 1:**  
2  
  
**Sample Input 2:**  
6  
52  
51  
81  
84  
60  
88  
  
**Sample Output 2:**  
71

**publicclass** Main {

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

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

**int**[] a= **newint**[n];

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

a[i]=sc.nextInt();

System.*out*.println(User.calculateMedian (a));

}

}

**publicclass** User {

**publicstaticint** calculateMedian(**int** s[]) {

**double** med=0;

**double** avg=0;

Arrays.*sort*(s);

**int** mid=s.length/2;

**if**(s.length%2!=0)

med=s[mid];

**else**

{

avg=(**double**)(s[mid]+s[mid-1]\)/2;

System.*out*.println(avg);

med=Math.*ceil*(avg);

}

**return**(**int**)med;

}

}

**43. Sequence in Array**

Write a program to accept an int array as input, and check if [1,2,3] appears somewhere in the same sequence.  
  
Include a class UserMainCode with a static method **searchSequence** which accepts the int array. The return type is a boolean which returns true or false.  
  
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 a an integer which denotes the size of the array followed by the array of integers.  
Output should print true or false.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
9  
11  
-2  
5  
1  
2  
3  
4  
5  
6  
  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
6  
-2  
5  
1  
3  
2  
6  
  
**Sample Output 2:**  
FALSE

**publicclass** Main {

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

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

**int**[] a= **newint**[n];

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

a[i]=sc.nextInt();

**boolean** b=User.*calculateMedian* (a);

System.*out*.println(b);

}

}

**publicclass** User {

**publicstaticboolean** calculateMedian(**int** s[]) {

**int**[] a={1,2,3};

**int** n=s.length-(a.length-1);

**boolean** b=**false**;

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

{

**if**(s[i]==a[0] )

{

**if**(s[i+1]==a[1])

{

**if**(s[i+2]==a[2])

{

b=**true**;

**break**;

}

**else**

b=**false**;

}

**else**

b=**false**;

}

**else**

b=**false**;

}

**return** b;

}

}

**44. Asterisk & Characters**

Write a program to read a string and return true or false based on the below rule:

1. Return true if for every '\*' in the string, there are same characters both side immediately before and after the star, else return false.  
  
Include a class UserMainCode with a static method **scanStarNeighbors** which accepts the string. The return type is the boolean TRUE or FALSE based on the rule.  
  
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 TRUE or FALSE.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
Hello\*World  
  
**Sample Output 1:**  
FALSE  
  
**Sample Input 2:**  
Welcome\*elizabeth  
  
**Sample Output 2:**  
TRUE

**publicclass** Main {

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

String n=sc.next();

**boolean** b=User.scanStarNeighbors(n);

System.*out*.println(b);

}

}

**publicclass** User {

**publicstaticboolean** scanStarNeighbors(String s) {

StringTokenizer st=**new** StringTokenizer(s,"\*");

**boolean** b=**false**;

**while**(st.hasMoreTokens())

{

String s1=st.nextToken();

String s2=st.nextToken();

**if**(s1.charAt(s1.length()-1)==s2.charAt(0))

{

b=**true**;

}

}

**return** b;

}

}

**45. Occurance Count**

Write a program to read a string that contains a sentence and read a word. Check the number of occurances of that word in the sentence.  
  
Include a class UserMainCode with a static method **countWords** which accepts the two strings. The return type is the integer giving the count.  
  
Note: The check is case-sensitive.  
  
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 count indicating the number of occurances.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
Hello world Java is best programming language in the world  
world  
  
**Sample Output 1:**  
2  
  
**Sample Input 2:**  
hello world  
World  
  
**Sample Output 2:**  
0

**publicclass** Main {

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

String s1=sc.nextLine();

String s2=sc.next();

**int**b=User.*countWords* (s1,s2);

System.*out*.println(b);

}

}

**publicclass** User {

**publicstaticint** countWords (String s1,String s2) {

StringTokenizer st=**new** StringTokenizer(s1," ");

**int** c=0;

**while**(st.hasMoreTokens())

{

String s3=st.nextToken();

**if**(s3.equals(s2))

{

c++;

}

}

**return** c;

}

}

**46.Regular Expressions - III**

Write a program to read two strings S1 & S2, compute the number of times that S2 appears in S1.  
  
Include a class UserMainCode with a static method **searchString** which accepts the two strings. The return type is the integer giving the count.  
  
Note: The check is case-insensitive.  
  
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 count indicating the number of occurances.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
Catcowcat  
cat  
  
**Sample Output 1:**  
2  
  
**Sample Input 2:**  
Catcowcat  
catp  
  
**Sample Output 2:**  
0

**publicclass** User {

**publicstaticint** scanStarNeighbors(String s1,String s2) {

**int** ls1=s1.length();

**int** ls2=s2.length();

**int** n=ls1-(ls2-1);

System.*out*.println(n);

**int** ct=0;

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

{

String ss=s1.substring(i,i+(ls2));

**if**(s2.equals(ss))

ct++;

}

**return** ct;

}

}

**47. Strings Processing**

Write a program to read a string that contains comma separated fruit names and also a number N. Pick the nth fruit and return it. If the total number of elements are less than the number specified in N, then return the last element.  
  
Include a class UserMainCode with a static method **findFruitName** which accepts the the string and the number n. The return type is the string which has the fruit name.  
  
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:**  
Apple,Banana,Orange  
2  
  
**Sample Output 1:**  
Banana  
  
**Sample Input 2:**  
Apple,Banana,Orange  
4  
  
**Sample Output 2:**  
Orange

**import** java.text.ParseException;

**import**java.util.ArrayList;

**import**java.util.HashMap;

**import**java.util.Iterator;

**import** java.util.Scanner;

**publicclass** Main {

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

String s1=sc.nextLine();

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

System.*out*.println(User.*findFruitName*(s1,n));

}

}

**import**java.util.ArrayList;

**import**java.util.Arrays;

**import**java.util.LinkedHashSet;

**import** java.util.StringTokenizer;

**publicclass** User {

**publicstatic** String findFruitName(String s1,**int** n) {

StringTokenizer st=**new** StringTokenizer(s1,",");

**int**c=0,i=0;

String ss=**null**;

String[] s=**new** String[st.countTokens()];

**while**(st.hasMoreTokens())

{

s[i]=st.nextToken();

i++;

}

**if**(i>n)

{

ss=s[n-1];

}

**else**

{

ss=s[i-1];//last element display

}

**return** ss;

}

}

**48. Proper Case**

Write a program to read a string and convert the intial letter of each word to uppercase.  
  
Include a class UserMainCode with a static method **changeCase** which accepts the string. The return type is the modified 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:**  
This is cognizant academy  
  
**Sample Output 1:**  
This Is Cognizant Academy

**49. Length of same word**

Write a program to read a string containing multiple words find the first and last words, if they are same, return the length and if not return the sum of length of the two words.  
  
Include a class UserMainCode with a static method **compareLastWords** which accepts the string. The return type is the length 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 a integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
This is Cognizant Academy  
  
**Sample Output 1:**  
11  
  
**Sample Input 2:**  
Hello World Hello  
  
**Sample Output 2:**  
5

**import** java.util.StringTokenizer;

**publicclass** UserMainCode {

**publicstatic** String changeWord(String s)

{

StringTokenizer st=**new** StringTokenizer(s," ");

StringBuffer sb=**new** StringBuffer();

**while**(st.hasMoreTokens())

{

String s1=st.nextToken();

sb.append(s1.substring(0,1).toUpperCase());

sb.append(s1.substring(1));

sb.append(" ");

}

**return** sb.toString();

}

}

**50. Perfect Number**

Write a program to that takes a positive integer andreturns true if the number is perfect number.  
  
A positive integer is called a perfect number if the sum of all its factors (excluding the number itself, i.e., proper divisor) is equal to its value.  
  
For example, the number 6 is perfect because its proper divisors are 1, 2, and 3, and 6=1+2+3; but the number 10 is not perfect because its proper divisors are 1, 2, and 5, and 1+2+5 is not equal to 10  
  
Include a class UserMainCode with a static method **getPerfection** which accepts the number. The return type is boolean (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 integer.  
Output consists of TRUE / FALSE.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
28  
  
**Sample Output 1:**  
TRUE

**publicclass** User {

**publicstaticboolean** scanStarNeighbors(**int**n) {

**boolean** b;

**int** sum=0;

**for**(**int** i=1;i<n;i++)

{

**if**(n%i==0)

{

sum=sum+i;

System.*out*.println(sum);

}

}

**if**(sum==n)

{

b=**true**;

}

**else**

{

b=**false**;

}

**return** b;

}

}  
-----------------------------------------

**51. Find Digits**

For a given double number with atleast one decimal value, Write a program to compute the number of digits before and after the decimal point in the following format – noOfDigitsBeforeDecimal:noOfDigitsAfterDecimal.  
  
Note: Ignore zeroes at the end of the decimal (Except if zero is the only digit after decimal. Refer Example 2 and 3)  
  
Include a class UserMainCode with a static method **findNoDigits** which accepts the decimal value. The return type is 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 double.  
Output consists of string.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
843.21  
  
**Sample Output 1:**  
3:2  
  
**Sample Input 2:**  
20.130  
  
**Sample Output 2:**  
2:2  
  
**Sample Input 3:**  
20.130  
  
**Sample Output 3:**  
2:2

import java.util.\*;

public class Main {

public static void main(String[] args) {

double d=845.69;

System.out.println(noOfDigits(d));

}

public static String noOfDigits(double d) {

int n1=0,n2=0;

String s=String.valueOf(d);

StringTokenizer t=new StringTokenizer(s,".");

String s1=t.nextToken();

String s2=t.nextToken();

n1=s1.length();

n2=s2.length();

if(s1.charAt(0)=='0')

n1=s1.length()-1;

if(n2!=1)

if(s2.charAt(s2.length()-1)=='0')

n2=s2.length()-1;

String s3=String.valueOf(n1)+":"+String.valueOf(n2);

return s3;

}

}

**import** java.util.StringTokenizer;

**publicclass** User{

**publicstatic** String noOfDigits(**double** d) {

**int** n1=0,n2=0;

String s=String.*valueOf*(d);

StringTokenizer t=**new** StringTokenizer(s,".");

String s1=t.nextToken();

String s2=t.nextToken();

n1=s1.length();

n2=s2.length();

**if**(s1.charAt(0)=='0')

n1=s1.length()-1;

//if(n2!=1)

**if**(s2.charAt(n2-1)=='0')

n2=s2.length()-1;

//String s3=String.valueOf(n1)+":"+String.valueOf(n2);

StringBuffer sb=**new** StringBuffer();

sb.append(n1).append(":").append(n2);

**return** sb.toString();

}

}

**52. Employees & Designations**

A Company wants to obtain employees of a particular designation. You have been assigned as the programmer to build this package. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:  
    Read Employee details from the User. The details would include name and designaton in the given order. The datatype for name and designation is string.  
    Build a hashmap which contains the name as key and designation as value.  
    You decide to write a function **obtainDesignation** which takes the hashmap and designation as input and returns a string List of employee names who belong to that designation as output. Include this function in class UserMainCode. Display employee name's in ascending order.  
Create a Class Main which would be used to read employee details in step 1 and build the hashmap. 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 two values indicate the employee name employee designation. The last string would be the designation to be searched.  
Output consists of a array values containing employee names.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
4  
Manish  
MGR  
Babu  
CLK  
Rohit  
MGR  
Viru  
PGR  
MGR  
  
**Sample Output 1:**  
Manish  
Rohit

class Main

{

public static void main(String[] arg)

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

sc.nextLine();

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

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

{

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

}

String b=sc.nextLine();

HashMap<String,String> op=new HashMap<String,String>();

op=MainClass.obtainDesig(hm,b);

Iterator<String> itr=op.keySet().iterator();

while(itr.hasNext())

{

String key=itr.next();

System.out.println(key);

String value=hm.get(key);

System.out.println(value);

}

}}

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;

public class MainClass {

public static HashMap<String,String> obtainDesig(HashMap<String,String> hm,String s)

{

LinkedHashMap<String,String> op=new LinkedHashMap<String,String>();

Iterator<String> itr=hm.keySet().iterator();

while(itr.hasNext())

{

String key=itr.next();

String value=hm.get(key);

if(s.equals(value))

{

op.put(key,value);

}

}

return op;

}

}

**53.Grade Calculator**

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

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Scanner;

public class Main

{

public static void main(String[]arg)

{

LinkedHashMap<String,Double>hm=new LinkedHashMap<String,Double>();

LinkedHashMap<String,String>hm1=new LinkedHashMap<String,String>();

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

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

{

String s=sc.next();

double d=sc.nextDouble();

hm.put(s,d);

}

LinkedHashMap<String,String>hm2=UserMainCode.dis(hm);

for(Map.Entry<String,String>entry:hm2.entrySet())

{

System.out.println(entry.getKey());

System.out.println(entry.getValue());

}}}

import java.util.LinkedHashMap;

import java.util.Map;

class UserMainCode

{

public static LinkedHashMap<String,String>dis(LinkedHashMap<String,Double>h1)

{

LinkedHashMap<String,String>h2=new LinkedHashMap<String,String>();

for(Map.Entry m:h1.entrySet())

{

double d=(Double)m.getValue();

if(d>60)

{

String s=(String)m.getKey();

h2.put(s,"pass");

}

else

{

String s=(String)m.getKey();

h2.put(s,"fail");

}

}

return h2;

}

}

(Or)

**import** java.util.\*;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

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

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

LinkedHashMap<String,Float> ip=**new** LinkedHashMap<String,Float>();

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

{

ip.put(sc.next(),sc.nextFloat());

}

LinkedHashMap<String,String> op=**new** LinkedHashMap<String,String>();

op=User.*noOfDigits*(ip);

Iterator<String> itr= op.keySet().iterator();

**while**(itr.hasNext())

{

String key=itr.next();

System.*out*.println(key);

String value=op.get(key);

System.*out*.println(value);

}

}}

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.LinkedHashMap;

**publicclass** User{

**publicstatic** LinkedHashMap<String,String> noOfDigits(HashMap<String,Float> hm) {

LinkedHashMap<String,String> op=**new** LinkedHashMap<String,String>();

Iterator<String> itr=hm.keySet().iterator();

String res=**null**;

**for**(**int** i=0;i<hm.size();i++)

{

**while**(itr.hasNext())

{

String key=itr.next();

**float** value=hm.get(key);

**if**(value>=60)

res="pass";

**else**

res="fail";

op.put(key,res);

}

}

**return** op;

}

}

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.LinkedHashMap;

**publicclass** User{

**publicstatic** LinkedHashMap<String,String> noOfDigits(HashMap<String,Float> hm) {

LinkedHashMap<String,String> op=**new** LinkedHashMap<String,String>();

Iterator<String> itr=hm.keySet().iterator();

String res=**null**;

**while**(itr.hasNext())

{

String key=itr.next();

**float** value=hm.get(key);

**if**(value>=60)

res="pass";

**else**

res="fail";

op.put(key,res);

}

**return** op;

}

}

**54. DOB - Validation**

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

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Scanner;

public class UserMainCode {

public static void main(String[] args)

{

String str=new String();

Scanner sc=new Scanner(System.in);

str=sc.nextLine();

SimpleDateFormat sdf=new SimpleDateFormat("MM/dd/yyyy");

sdf.setLenient(false);

try

{

Date d1=sdf.parse(str);

System.out.println("TRUE");

}

catch(Exception e)

{

System.out.println("FALSE");

}

}

}

**55. Experience Validator**

Write a program to valiadate the experience of an employee.  
    An employee who has recently joined the organization provides his year of passing and total number of years of experience in String format. Write code to validate his experience against the current date.  
  
1) Input consists of two String first represent the year of passed out and the second string reperesent the year of experience.  
2) create a function with  name **validateExp**which accepts two string as input and boolean as output.  
3) The difference between current year and year of pass should be more than or equal to Experience  
Return true if all condition are true.  
  
Note:Conside 2015 as the current year.  
  
Include a class UserMainCode with the static function validateExp  
  
Create a Class Main which would be used to accept the boolean and call the static method present in UserMainCode.  
  
**Input and Output Formate:**  
Input consists of two Strings.  
output will display true if the given data are correct.  
  
**Sample Input:**  
2001  
5  
  
**Sample Output:**  
TRUE

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class Main {

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

String s=sc.nextLine();

String s1=sc.nextLine();

System.out.print(UserMainCode.getvalues(s,s1));

}

}

import java.util.Calendar;

import java.util.Date;

public class UserMainCode {

public static boolean getvalues(String s,String s1)

{

int y1=Integer.parseInt(s);

Date d=new Date();

Calendar c=Calendar.getInstance();

int y2=c.get(Calendar.YEAR);

int y=Math.abs(y1-y2);

int e=Integer.parseInt(s1);

if(y>=e)

return true;

else

return false;

}}

**56. ArrayList to String Array**

Write a program that performs the following actions:  
    Read n strings as input.  
    Create an arraylist to store the above n strings in this arraylist.  
    Write a function convertToStringArray which accepts the arraylist as input.  
    The function should sort the elements (strings) present in the arraylist and convert them into a string array.  
    Return the array.  
Include a class UserMainCode with the static method **convertToStringArray** which accepts an arraylist and returns an array.  
  
Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of n+1 integers. The first integer denotes the size of the arraylist, the next n strings are values to the arraylist.  
Output consists of an arrayas per step 4.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
4  
a  
d  
c  
b  
**Sample Output 1:**  
a  
b  
c  
d

import java.util.ArrayList;

import java.util.Scanner;

class Main

{

public static void main(String[] arg)

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

sc.nextLine();

ArrayList<String> aa=new ArrayList<String>();

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

{

aa.add(sc.nextLine());

}

String a[]=MainClass.convertToString( aa);

for(int i=0;i<a.length;i++)

{

System.out.println(a[i]);

}

}}

import java.util.ArrayList;

import java.util.Collections;

public class MainClass {

public static String[] convertToString(ArrayList<String> a1)

{

Collections.sort(a1);// uses to sort arraylist string

String a[]=new String[a1.size()];

a1.toArray(a);

return a;

}

}

**57. State ID generator**

Write a program to generate the state ID.  
     1)Read n Strings as input(as State Name).  
     2)Create a String Array to Store the above Input.  
     3)Write a function **getStateId** which accepts String Array as input.  
     4)Create a HashMap<String,String> which stores state name as key and state Id as Value.  
     5)The function getStateId returns the HashMap to the Main Class.  
  
Include UserMainCode Class With static method **getStateId** which accepts String array and return a hashmap.  
  
Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.  
  
  
**Input and Output Format:**  
Input Consists of an integer n denotes the size of the string array.  
Output consists of an HashMap displayed in the string array order.  
  
**Sample Input 1:**  
3  
Kerala  
Gujarat  
Goa  
  
**Sample Output 1:**  
KER:Kerala  
GUJ:Gujarat  
GOA:Goa

**import** java.util.\*;

**publicclass** Main

{

**publicstaticvoid** main(String[] args)

{

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

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

String s1[]=**new** String[n];

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

{

s1[i]=sc.next();

}

LinkedHashMap<String,String>ip=**new** LinkedHashMap<String,String>();

ip=User.*Method*(s1);

Iterator<String> itr=ip.keySet().iterator();

//while(itr.hasNext())

**for**(**int** i=0;i<ip.size();i++)

{

String key=itr.next();

String value=ip.get(key);

System.*out*.println(value+""+key);

}

}

}

**import** java.util.LinkedHashMap;

**publicclass** User

{

**publicstatic** LinkedHashMap<String,String> Method(String[] s1)

{

LinkedHashMap<String,String> op=**new** LinkedHashMap<String,String>();

**for**(**int** i=0;i<s1.length;i++)

{

StringBuffer sb=**new** StringBuffer();

StringBuffer key=sb.append(s1[i].substring(0,3)).append(":");

op.put(s1[i],key.toString().toUpperCase());

}

**return** op;

}

}

(or)

STATE id

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.HashMap;

public class Main {

public static void main(String[] args) {

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

try

{

int n=Integer.parseInt(br.readLine());

String[] input=new String[n];

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

{

input[i]=br.readLine();

}

HashMap<String,String>hm=UserMainCode.costEstimator(input);

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

{

String s=input[i];

String key=hm.get(s);

System.out.println(key+":"+s);

}

}

catch(Exception e)

{

}

}

}

import java.util.HashMap;

public class UserMainCode {

public static HashMap<String,String> costEstimator(String[] name)

{

int n=name.length;

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

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

{

String sub=name[i].substring(0, 3);

hm.put(name[i],sub.toUpperCase());

}

return hm;

}

}

**58.ArrayList to String Array**

Write a program that performs the following actions:  
  
1.Read m strings as input (fruit names).  
2.Create an arraylist to store the above m strings in this arraylist.  
3.Read n strings as input (fruit names).  
4.Create an arraylist to store the above n strings in this arraylist.  
5.Write a function fruitSelector which accepts the arraylists as input.  
6.Remove all fruits whose name ends with 'a' or 'e' from first arrayList and remove all fruits whose name begins  with 'm' or 'a' from second arrayList then combine the two lists and return the final output as a String array.  
7.If the array is empty the program will print as “No fruit found”  
Include a class UserMainCode with the static method **fruitSelector** which accepts the two arraylists and returns an array.  
  
Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.  
  
Output consists of an array as per step 6. Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
3  
Apple  
Cherry  
Grapes  
4  
Orange  
Mango  
Melon  
Apple  
**Sample Output 1:**  
Cherry  
Grapes  
Orange

import java.util.\*;

public class Main {

public static void main(String[] args) {

List<String> l1=new ArrayList<String>();

l1.add("Apple");

l1.add("Chery");

l1.add("Grapes");

List<String> l2=new ArrayList<String>();

l2.add("Orange");

l2.add("Mango");

l2.add("Melon");

l2.add("Apple");

String[] s2=fruitsList(l1,l2);

for(String s3:s2)

System.out.println(s3);

}

public static String[] fruitsList(List<String> l1, List<String> l2){

List<String> l3=new ArrayList<String>();

for(int i=0;i<l1.size();i++){

String s1=l1.get(i);

if(s1.charAt(s1.length()-1)!='a' && s1.charAt(s1.length()-1)!='A' && s1.charAt(s1.length()-1)!='e' && s1.charAt(s1.length()-1)!='E')

l3.add(s1); }

for(int i=0;i<l2.size();i++){

String s1=l2.get(i);

if(s1.charAt(0)!='m' && s1.charAt(0)!='M' && s1.charAt(0)!='a' && s1.charAt(0)!='A')

l3.add(s1); }

Collections.sort(l3);

String[] s2=new String[l3.size()];

for(int i=0;i<s2.length;i++)

s2[i]=l3.get(i);

return s2;

}

**import** java.util.\*;

**publicclass** Main {

**publicstaticvoid** main(String[] args) {

List<String> l1=**new** ArrayList<String>();

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

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

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

{

l1.add(sc.next());

}

List<String> l2=**new** ArrayList<String>();

**int** n1=sc.nextInt();

**for**(**int** i=0;i<n1;i++)

{

l2.add(sc.next());

}

String[] s2=User.*fruitsList*(l1,l2);

// for(int i=0;i<s2.length;i++)

// System.out.println(s2[i].toString());

**for**(String s3:s2)

System.*out*.println(s3);

}

}

**publicclass** User

{

**publicstatic** String[] fruitsList(List<String> l1, List<String> l2){

ArrayList<String> l3=**new** ArrayList<String>();

**for**(**int** i=0;i<l1.size();i++)

{

String s1=l1.get(i);

**int**len=s1.length();

**if**(s1.charAt(len-1)!='a'&& s1.charAt(len-1)!='A'

&& s1.charAt(len-1)!='e'&& s1.charAt(len-1)!='E')

l3.add(s1);

}

**for**(**int** i=0;i<l2.size();i++)

{

String s1=l2.get(i);

**if**(s1.charAt(0)!='m'&& s1.charAt(0)!='M'&& s1.charAt(0)!='a'

&& s1.charAt(0)!='A')

l3.add(s1);

}

Collections.*sort*(l3);

String[] s2=**new** String[l3.size()];

**for**(**int** i=0;i<s2.length;i++)

s2[i]=l3.get(i);

**return** s2;

}

}

]

/’

**59. Elements in ArrayList**

Use Collection Methods.  
Write a program that takes two ArrayLists as input and  finds out all elements present either in A or B, but not in both.

Include a class UserMainCode with the static method arrayListSubtractor which accepts the two arraylists and returns an array.  
  
Create a Class Main which would be used to read the inputs and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.  
  
Output consists of an array. The elements in the output array need to be printed in sorted order.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
4  
1  
8  
3  
5  
2  
3  
5  
**Sample Output 1:**  
1  
8  
  
**Sample Input 2:**  
4  
9  
1  
3  
5  
4  
1  
3  
5  
6  
**Sample Output 2:**  
6  
9

**import** java.util.\*;

**publicclass** Main

{

**publicstaticvoid** main(String[] args)

{

**int** n,m;

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

n = sc.nextInt();

ArrayList<Integer> a1 = **new** ArrayList<Integer>();

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

{

a1.add(sc.nextInt());

}

m = sc.nextInt();

ArrayList<Integer> a2 = **new** ArrayList<Integer>();

**for**(**int** i=0;i<m;i++)

{

a2.add(sc.nextInt());

}

**int**[] result = User.*arrayListSubtractor*(a1, a2);

Arrays.*sort*(result);

**for**(**int** i=0;i<result.length;i++)

System.*out*.println(result[i]);

}

}

**import** java.util.\*;

**publicclass** User

{

**publicstaticint**[] arrayListSubtractor(ArrayList<Integer>

arrlist1,ArrayList<Integer> arrlist2)

{

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<arrlist1.size();i++)

ts1.add(arrlist1.get(i));

**for**(**int** i=0;i<arrlist2.size();i++)

ts2.add(arrlist2.get(i));

ts1.addAll(ts2);

**for**(**int** i=0;i<arrlist1.size();i++)

{

**for**(**int** j=0;j<arrlist2.size();j++)

{

**if**(arrlist1.get(i)==arrlist2.get(j))

ts3.add(arrlist1.get(i));

}

}

ts1.removeAll(ts3);

aa.addAll(ts1);

**int** res[]=**newint**[aa.size()];

**for**(**int** i=0;i<res.length;i++)

res[i]=aa.get(i);

**return** res;

}

}

**60. Price Calculator - II**

Write a small price calculator application with the below mentioned flow:  
  
1. Read a value n indicating the total count of devices. This would be followed by the name and price of the device. The datatype for name would be String and price would be float.  
  
2. Build a hashmap containing the peripheral devices with name as key and price as value.  
  
3. Read a value m indicating the number of devices for which the price has to be calculated. This would be followed by device names.  
  
4. For each devices mentioned in the array calcuate the total price.  
  
5. You decide to write a function costEstimator which takes the above hashmap and array as input and returns the total price (float) as output with two decimal points. Include this function in class UserMainCode.  
  
Create a Class Main which would be used to read details in step 1 and build the hashmap. Call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of device details. The first number indicates the size of the devices. The next two values indicate the name,price.  
  
This would be followed by m indicating the size of the device array. The next m values would be the device names.  
Output consists of the total price in float.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
3  
Monitor  
1200.36  
Mouse  
100.42  
Speakers  
500.25  
2  
Speakers  
Mouse  
**Sample Output 1:**  
600.67

import java.util.\*;

public class UserMainCode {

public static void main(String[] args) {

HashMap<String, String> m1=new HashMap<String, String>();

m1.put("monitor", "1200.36");

m1.put("mouse","100.42");

m1.put("speaker", "500.25");

String[] s={"speaker","mouse"};

System.out.println(getTheTotalCostOfPheripherals(m1,s));

}

public static float getTheTotalCostOfPheripherals(HashMap<String,String> m1,String[] s) {

Float f=(float) 0;

Iterator<String> i=m1.keySet().iterator();

while(i.hasNext()){

String s1=(String) i.next();

Float f1=Float.parseFloat(m1.get(s1));

for(int j=0;j<s.length;j++)

if(s[j].equals(s1))

f+=f1;

}

return f;

}}

**61.String Processing - ZigZag**

Write a program to read a string containing date in DD-MM-YYYY format. find the number of days in the given month.  
  
Note - In leap year February has got 29 days.  
  
Include a class UserMainCode with a static method **getLastDayOfMonth** which accepts the string. The return type is the integer having number of days.  
  
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:**  
12-06-2012  
**Sample Output 1:**  
30  
  
**Sample Input 2:**  
10-02-2012  
**Sample Output 2:**  
29  
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

public class User {

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

// TODO Auto-generated method stub

String s1="10-02-2012";

SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");

Calendar cal=Calendar.getInstance();

Date d1=sdf.parse(s1);

cal.setTime(d1);

int n=cal.getActualMaximum(Calendar.DAY\_OF\_MONTH);

System.out.println(n);

}

}

**62. Leap Year**

Write a program to read a string containing date in DD/MM/YYYY format and check if its a leap year. If so, return true else return false.  
  
Include a class UserMainCode with a static method **isLeapYear** which accepts the string. The return type is the boolean indicating 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:**  
  
23/02/2012  
  
**Sample Output 1:**  
  
TRUE  
  
**Sample Input 2:**  
  
12/12/2011  
  
**Sample Output 2:**  
  
FALSE

**import** java.text.ParseException;

**import** java.util.\*;

**publicclass** Main

{

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

String s=sc.nextLine();

System.*out*.println(User.*leapYear*(s));

}

}

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.\*;

**publicclass** User

{

**publicstaticboolean** leapYear(String s) **throws** ParseException

{

SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");

GregorianCalendar g=**new** GregorianCalendar();

Calendar cal=Calendar.*getInstance*();

Date d1=sdf.parse(s);

cal.setTime(d1);

**int**n=cal.get(Calendar.*YEAR*);

**boolean** b=g.isLeapYear(n);

**return** b;

}}

**63. Largest Chunk**

Write a program to read a string and return the length of the largest "chunk" in the string.  
A chunk is a repetition of same character 2 or more number of times. If the given string doest not contain any repeated chunk of characters return -1.  
Include a class UserMainCode with a static method **getLargestSpan** which accepts the string. The return type is the 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 string.  
Output consists of integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
This place is soooo good  
**Sample Output 1:**  
4

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);

}

}

**64. Largest Span**

Write a program to read a integer array, find the largest span in the array.  
Span is the count of all the elements between two repeating elements including the repeated elements.  
Include a class UserMainCode with a static method **getLargestSpan** 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:**  
6  
4  
2  
1  
4  
5  
7  
**Sample Output 1:**  
4

public class Main {

public static void main(String[] args) {

int[]a={1,2,1,1,3};

System.out.println(maxSpan(a));

}

public static int maxSpan(int[] a) {

String s2 = null;

int n=0;

StringBuffer sb=new StringBuffer();

for(int i=0;i<a.length;i++)

sb.append(String.valueOf(a[i]));

String s1=sb.toString();

for(int i=0;i<s1.length();i++)

for(int j=i+1;j<s1.length();j++)

if(s1.charAt(i)==s1.charAt(j))

s2=String.valueOf(s1.charAt(j));

int n1=s1.indexOf(s2);

int n2=s1.lastIndexOf(s2);

for(int i=n1+1;i<n2;i++)

n++;

return (n+2);

}

}

**65.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

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.LinkedHashSet;

import java.util.Scanner;

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.display(a));

}}

import java.util.Iterator;

import java.util.LinkedHashSet;

public class UserMainCode {

public static int display(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;

}}

**66.Regular Expression - III**

Given a string (s)  apply the following rules.  
I)At least 8 characters must be present  
II)At least one capital letter must be present  
III)At least one small letter must be present  
Iv)At least one special symbol must be present  
V)At least one numeric value must be present  
If the condition is satisifed then print valid else print invalid.  
  
Include a class UserMainCode with a static method **passwordValidation** 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 string (valid / invalid) .  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
Technology$1213  
**Sample Output 1:**  
valid

public class UserMainCode

{

public static int display(String s)

{

if(s.matches(".\*[0-9]{1,}.\*") && s.matches(".\*[@#$]{1,}.\*") && s.length()>=8 && s.matches(".\*[A-Z]{1,}.\*") && s.matches(".\*[a-z]{1,}.\*"))

return 1;

else

return -1;

}}

**import** java.util.\*;

**publicclass** Main

{

**publicstaticvoid** main(String[] args) {

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

String s=sc.nextLine();

System.*out*.println(User.*leapYear*(s));

}

}

**publicclass** User{

**publicstaticint** leapYear(String s)

{

**if**(s.matches(".\*[0-9]{1,}.\*")

&&s.matches(".\*[!@#$%^&\*]{1,}.\*") &&s.length()>=8 &&

s.matches(".\*[A-Z]{1,}.\*") &&s.matches(".\*[a-z]{1,}.\*"))

**return** 1;

**else**

**return** -1;

}}

**67.Integer Factorial**

Give an array of integer as input, store the numbers and their factorials in an hashmap and print the same.  
Include a class UserMainCode with a static method **getFactorial** which accepts the integer array. The return type is the hashmap which is printed key:value.  
  
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 a number denoting the size of the array and followed by the elements.  
Output consists of a hashmap printed in the output format .  
Refer sample output for formatting specifications.  
  
**Sample Input1:**  
4  
2  
3  
5  
4  
**Sample Output1:**  
2:2  
3:6  
5:120  
4:24

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;

import java.util.Scanner;

public class kapes3 {

public static void main(String []args){

Scanner sc=new Scanner(System.in);

int s=Integer.parseInt(sc.nextLine());

int []a=new int[s];

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

{

a[i]=sc.nextInt();

}

LinkedHashMap<Integer,Integer>hm2=new LinkedHashMap<Integer,Integer>();

hm2=kapes4.display(a);

Iterator<Integer> it=hm2.keySet().iterator();

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

{

int n=it.next();

int fac=hm2.get(n);

System.out.println(n+":"+fac);

}

}

}

import java.text.DecimalFormat;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;

public class kapes4

{public static LinkedHashMap<Integer,Integer> display(int[] a)

{

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

for(int i=0;i<a.length;i++)

{

int u=1;

for(int j=1;j<=a[i];j++)

{

u=u\*j;

}

hm.put(a[i],u);

}

return hm;

}}

**68. String processing – Long + Short + Long**

Obtain two strings S1,S2 from user as input. Your program should form a string of  “long+short+long”, with the shorter string inside of the longer String.  
Include a class UserMainCode with a static method **getCombo** which accepts two string variables. The return type is the 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 two strings with maximum size of 100 characters.  
Output consists of an string.  
  
Refer sample output for formatting specifications.  
**Sample Input 1:**  
Hello  
Hi  
**Sample Output 1:**  
HelloHiHello

import java.util.StringTokenizer;

public class User {

public static void main(String[] args){

String s1="Hi";

String s2="Hello";

System.out.println(capsStart(s1,s2));

}

public static String capsStart(String s1,String s2){

StringBuffer s5=new StringBuffer();

int q=s1.length();

int w=s2.length();

if(q>w)

{

s5.append(s1).append(s2).append(s1);

}

else

{

s5.append(s2).append(s1).append(s2);

}

return s5.toString();

}

}

**69. Age for Voting**

Given a date of birth (dd/MM/yyyy) of a person in string, compute his age as of 01/01/2015.  
  
If his age is greater than 18, then println eligible else println not-eligible.  
  
Include a class UserMainCode with a static method getAge which accepts the string value. 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:**  
16/11/1991  
  
**Sample Output 1:**  
eligible

**import** java.util.\*;

**publicclass** Main

{

**publicstaticvoid** main(String[] args) {

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

String s =sc.nextLine();

System.*out*.println(User.*display*(s));

}}

**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

**import** java.util.Date;

**publicclass** User{

**publicstatic** String display(String n)

{

**int** year=0;

String now="01/01/2015";

SimpleDateFormat sdf1=**new** SimpleDateFormat("dd/MM/yyyy");

**try**

{

sdf1.setLenient(**false**);

Calendar c1=Calendar.*getInstance*();

Date d=sdf1.parse(n);

c1.setTime(d);

**int** y=c1.get(Calendar.*YEAR*);

**int** m=c1.get(Calendar.*MONTH*);

**int** day=c1.get(Calendar.*DAY\_OF\_MONTH*);

Calendar c2=Calendar.*getInstance*();

Date d1=sdf1.parse(now);

c1.setTime(d1);

**int** y1=c2.get(Calendar.*YEAR*);

**int** m1=c2.get(Calendar.*MONTH*);

**int** day1=c2.get(Calendar.*DAY\_OF\_MONTH*);

year=y1-y;

//System.out.println(year);

**if**(m>m1)

year--;

**elseif**(m==m1)

{**if**(day<day1)

year--;

}

}

**catch**(Exception e)

{

e.printStackTrace();

}

**if**(year>18)

**return**"eligible";

**else**

**return**"not-eligible";

}}

**public** **class** UserMainCode{

**public** **static** **int** getMaxSpan(**int** a[]) {

**int** i,j,k,count,max=0,p=0;

**int** n=a.length;

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

            {

                  count=0;

**for**(j=i+1;j<n;j++)

                  {

**if**(a[i]==a[j])

                        {

                              p=j;

                        }

                  }

**for**(k=i;k<=p;k++)

                  {

                        count++;

                  }

**if**(count>max)

                  {

                        max=count;

                  }

            }

**return** max;

            }

}