**2.**    **Number Validation**

Write a program to read a string of 10 digit number , check whether the string contains a 10 digit number in the format XXX-XXX-XXXX where 'X' is a digit.

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

The return type of the output should be 1 if the string meets the above specified format . In case the number does not meet the specified format then return -1 as output.

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

**Input and Output Format:**

Input consists of a string.

Output is a string specifying the given string is valid or not .

Refer sample output for formatting specifications.

**Sample Input 1:**

123-456-7895

**Sample Output 1:**

Valid number format

**Sample Input 2:**

-123-12344322

**Sample Output 2:**

Invalid number format

**2.**    **Number Validation**

**import**java.util.Scanner;

**publicclass**UserMainCode {

**staticint**validateNumber(String s1)

{

**if**(s1.matches("[0-9]{3}[-]{1}[0-9]{3}[-]{1}[0-9]{4}"))

{

**return** 1;

}

**else**

**return** -1;

}

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

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

**int** c=UserMainCode.*validateNumber*(n1);

**if**(c==1)

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

**else**

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

}

}

**4.**    **Fetching Middle Characters from String**

Write a program to read a string of even length and to fetch two middle most characters from the input string and return it as string output.

Include a class **UserMainCode** with a static method **getMiddleChars** which accepts a string of even length as input . The return type is a string which should be the middle characters of the string.

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

**Input and Output Format:**

Input consists of a string of even length.

Output is a string .

Refer sample output for formatting specifications.

**Sample Input 1:**

this

**Sample Output 1:**

hi

**Sample Input 1:**

Hell

**Sample Output 1:**

el

**4.**    **Fetching Middle Characters from String**

**import**java.util.Scanner;

**publicclass**UserMainCode {

/\*\*

\* **@param**args

\*/

**staticvoid**getMiddleChars(String s1)

{

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

**int** mid=k/2;

StringBuffersb1= **new**StringBuffer();

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

{

sb1.append(s1.charAt(mid-1));

sb1.append(s1.charAt(mid));

}

System.*out*.println(sb1);

}

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

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

UserMainCode.*getMiddleChars*(n1);

}

}

**5.**    **Check Characters in a String**

Write a program to read a string  and to test whether first and last character are same. The string is said to be be valid if the 1st and last character are the same. Else the string is said to be invalid.

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

The return type of this method is an int.  Output should be 1 if the first character and last character are same . If they are different then return -1 as output.

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

**Input and Output Format:**

Input consists of a string.

Output is a string saying characters are same or not .

Refer sample output for formatting specifications.

**Sample Input 1:**

the picture was great

**Sample Output 1:**  
Valid

**Sample Input 1:**

this

**Sample Output 1:**

Invalid

**5.**    **Check Characters in a String**

**import** java.util.Scanner;

**class** Main {

/\*\*

\* **@paramargs**

\*/

**public** **static** **void** checkCharacters(String s1)

{

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

**char** c=s1.charAt(0);

**char** d=s1.charAt(k-1);

**if**(c==d)

{

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

}

**else**

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

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

Main.*checkCharacters*(n1);

}

}

**6.**    **Forming New Word from a String**

Write a program to read a string and a positive integer n as input and construct a string with first n and last n characters in the given string.

Include a class **UserMainCode** with a static method **formNewWord** which accepts a string and positive integer .

The return type of the output should be a string (value) of first n character and last n character.

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

**Input and Output Format:**

Input consists of a string of even length.

Output is a string .

Note: The given string length must be >=2n.

Refer sample output for formatting specifications.

**Sample Input 1:**

California

3

**Sample Output 1:**

Calnia

**Sample Input 2:**

this

1

**Sample Output 2:**

Ts

**6.**    **Forming New Word from a String**

**Import** java.util.Scanner;

**Public class** UserMainCode {

/\*\*

\* **@param**args

\*/

**Static void** checkCharacters(String s1,**int** a)

{

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

StringBuffer sb1= **new** StringBuffer();

**if**((2\*a)<=k)

{

sb1.append(s1.substring(0,a));

sb1.append(s1.substring(k-a));

}

System.*out*.println(sb1);

}

**Public static void** main(String[] args) {

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

**Int** num=in.nextInt();

UserMainCode.*checkCharacters*(n1,num);

}

}

**11.**  **String Encryption**

Given an input as string and write code to encrypt the given string using following rules and return the encrypted string:

1. Replace the characters at odd positions by next character in alphabet.

2. Leave the characters at even positions unchanged.

Note:

- If an odd position charater is 'z' replace it by 'a'.

- Assume the first character in the string is at position 1.

Include a class **UserMainCode** with a static method **encrypt** which accepts a string.

The return type of the output is the encrypted string.

Create a **Main** class which gets string as an input and call the static method **encrypt** present in the**UserMainCode.**

**Input and Output Format:**

Input is a string .

Output is a string.

**Sample Input 1:**

curiosity

**Sample Output 1:**

dusipsjtz

**Sample Input 2:**

zzzz

**Sample Output 2:**

Azaz

**11.**  **String Encryption**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**static** **void** checkCharacters(String s1)

{

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

StringBuffer sb1= **new** StringBuffer();

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

{

**char** d=s1.charAt(i);

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

{

**if**(d=='z')

{

sb1.append('a');

}

**else**

{

**int** c=(**int**)d;

++c;

sb1.append((**char**)c);

}

}

**else**

{

sb1.append(d);

}

}

System.*out*.println(sb1);

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

UserMainCode.*checkCharacters*(n1);

}

}

**12.**  **Password Validation**

Given a method with a password in string format as input. Write code to validate the password using following rules:

- Must contain at least one digit

- Must contain at least one of the following special characters @, #, $

# Length should be between 6 to 20 characters.

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

If the password is as per the given rules return 1 else return -1.If the return value is 1 then print valid password else print as invalid password.

Create a **Main** class which gets string as an input and call the static method **validatePassword** present in the **UserMainCode.**

**Input and Output Format:**

Input is a string .

Output is a string .

**Sample Input 1:**

%Dhoom%

**Sample Output 1:**  
Invalid password

**Sample Input 2:**

#@6Don

**Sample Output 2:**

Valid password

**12.**  **Password Validation**

**import** java.util.Scanner;

**public class** UserMainCode {

s**tatic void** checkCharacters(String s1)

{

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

**if**((k>=6 && k<=20) && s1.matches(".\*[0-9]{1}.\*") &&s1.matches(".\*[@$#]{1}.\*")) // .\* for some values will come in its place

// .\* similar to % in SQL

{

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

}

**else**

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

}

**Public static void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner in=**new s**canner(System.*in*);

String n1=in.nextLine();

UserMainCode.*checkCharacters*(n1);

}

}

**13.**  **Removing vowels from String**

Given a method with string input. Write code to remove vowels from even position in the string.

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

The return type of the output is string after removing all the vowels.

Create a **Main** class which gets string as an input and call the static method **removeEvenVowels** present in the **UserMainCode.**

**Input and Output Format:**

Input is a string .

Output is a string .

Assume the first character is at position 1 in the given string.

**Sample Input 1:**

commitment

**Sample Output 1:**

cmmitmnt

**Sample Input 2:**

capacity

**Sample Output 2:**

Cpcty

**13.**  **Removing vowels from String**

**import** java.util.Scanner;

**class** UserMainCode {

**public static** String removeEvenElements(String s1)

{

StringBuffer sb1=**new** StringBuffer();

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

{

**char** k=s1.charAt(i);

**if**((i+1)%2!=0)

{

sb1.append(k);

}

**else**

{

**if**(k!='a'&& k!='e'&& k!='i'&& k!='o'&& k!='u'&& k!='A'&& k!='E'&& k!='I'&& k!='O'&& k!='U')

{

sb1.append(k);

}

}

}

**return** sb1.toString();

}

**public static void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(*removeEvenElements*(s1));

in.close();

}

}

**18.Calculate Electricity Bill**

Given a method calculateElectricityBill() with three inputs. Write code to calculate the current bill.

Include a class **UserMainCode** with a static method **calculateElectricityBill** which accepts 3 inputs .The return type of the output should be an integer .

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

**Input and Output Format:**

Input consist of 3 integers.

First input is previous reading, second input is current reading and last input is per unit charge.

Reading Format - XXXXXAAAAA where XXXXX is consumer number and AAAAA is meter reading.

Output is a single integer corresponding to the current bill.

Refer sample output for formatting specifications.

**Sample Input 1:**

ABC2012345

ABC2012660

4

**Sample Output 1:**

**1260**

**Sample Input 2:**

ABCDE11111

ABCDE11222

3

**Sample Output 2:**

333

**18.Calculate Electricity Bill**

**Import** java.util.Scanner;

**Public class** UserMainCode {

**Static int** calculateElectricityBill(String str1,String str2,**int** unit)

{

**int** e1=Integer.*parseInt*(str1.substring(5));

**int** e2=Integer.*parseInt*(str2.substring(5));

**int** total=(e2-e1)\*unit;

**return** total;

}

**Public static void** main(String args[])

{

String s1,s2;

**int** n;

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

s1=in.nextLine();

s2=in.nextLine();

n=in.nextInt();

**int** ans=UserMainCode.*calculateElectricityBill*(s1,s2,n);

System.*out*.println(ans);

}

}

**19.Sum of Digits in a String**

Write code to get the sum of all the digits present in the given string.

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

Return the sum as output. If there is no digit in the given string return -1 as output.

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

**Input and Output Format:**

Input consists of a string.

Output is a single integer which is the sum of digits in a given string.

Refer sample output for formatting specifications.

**Sample Input 1:**

good23bad4

**Sample Output 1:**

9

**Sample Input 2:**

good

**Sample Output 2:**

-1

**19.Sum of Digits in a String**

**import** java.util.Scanner;

**public clas** UserMaincode

{

s**tatic int** sumOfDigits(String str)

{

**int** k=str.length();

**int** sum=0,i;

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

{

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

{

StringBuffer s=**new** StringBuffer();

s.append(str.charAt(i));

sum=sum+Integer.*parseInt*(str.charAt(i)+””);

}

}

**if**(sum>=0)

**return** sum;

**else**

**return** -1;

}

**public static void** main(String args[])

{

String s1;

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

s1=in.nextLine();

**int** ans=UserMaincode.*sumOfDigits*(s1);

System.*out*.println(ans);

}

}

**20.String Concatenation**

Write code to get two strings as input and If strings are of same length simply append them together and return the final string. If given strings are of different length, remove starting characters from the longer string so that both strings are of same length then append them together and return the final string.

Include a class **UserMainCode** with a static method **concatstring** which accepts two string input.

The return type of the output is a string which is the concatenated string.

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

**Input and Output Format:**

Input consists of two strings.

Output is a string.

Refer sample output for formatting specifications.

**Sample Input 1:**

Hello

hi

**Sample Output 1:**

lohi

**Sample Input 2:**

Hello

Delhi

**Sample Output 2:**

HelloDelhi

**20.String Concatenation**

**import**java.util.Scanner;

**public class** UserMainCode {

/\*\*

\* **@param**args

\*/

**static** String concatstring(String s1,String s2)

{

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

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

String s3="0";

**if**(k==a)

{

s3=s1.concat(s2);

}

**if**(k>a)

{

s3=(s1.substring(k-a).concat(s2));

}

**if**(k<a)

{

s3=((s1.concat(s2.substring(a-k))));

}

**return** s3;

}

**public static void** main(String[] args) {

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

String n2=in.nextLine();

String n3=UserMainCode.*concatstring*(n1,n2);

System.*out*.println(n3);

}

}

**21. Color Code**

Write a program to read a string and validate whether the given string is a valid color code based on the following rules:

- Must start with "#" symbol

- Must contain six characters after #

- It may contain alphabets from A-F or digits from 0-9

Include a class **UserMainCode** with a static method **validateColorCode** which accepts a string. The return type (integer) should return 1 if the color is as per the rules else return -1.

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 (Valid or Invalid).

Refer sample output for formatting specifications.

**Sample Input 1:**

#FF9922

**Sample Output 1:**  
Valid

**Sample Input 2:**

#FF9(22

**Sample Output 2:**

Invalid

**21. Color Code**

**import**java.util.Scanner;

**publicclass**UserMainCode {

**publicstaticint**validateColorCode(String s1)

{

**if**(s1.matches("[#]{1}[A-F|0-9]{6}"))

**return** 1;

**else**

**return** -1;

}

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

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

String s1=in.nextLine();

**int** k=UserMainCode.*validateColorCode*(s1);

**if**(k==1)

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

**else**

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

in.close();

}

}

**22.Three Digits**

Write a program to read a string and check if the given string is in the format "CTS-XXX" where XXX is a three digit number.

Include a class **UserMainCode** with a static method **validatestrings** which accepts a string. The return type (integer) should return 1 if the string format is correct else return -1.0

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 (Valid or Invalid).

Refer sample output for formatting specifications.

**Sample Input 1:**

CTS-215

**Sample Output 1:**

Valid

**Sample Input 2:**

CTS-2L5

**Sample Output 2:**

Invalid

**22.Three Digits**

**import**java.util.Scanner;

**publicclass**UserMainCode {

**publicstaticint**validatestrings(String s1)

{

**if**(s1.matches("(CTS)[-]{1}[0-9]{3}")) // () anything typed within this brackets is a kind of constant which is displayed as such;

**return** 1;

**else**

**return** -1;

}

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

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

String s1=in.nextLine();

**int** k=UserMainCode.*validatestrings*(s1);

**if**(k==1)

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

**else**

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

in.close();

}

}

**27.Validating Input Password**

102.Write a and validate using the rules specified below. Apply following validations:

1. Minimum length should be 8 characters

2. Must contain any one of these three special characters @ or \_ or #

3. May contain numbers or alphabets.

4. Should not start with special character or number

5. Should not end with special character

Include a class **UserMainCode** with a static method **validatePassword** which accepts password string as input and returns an integer. The method returns 1 if the password is valid. Else it returns -1.

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

**Input and Output Format:**

Input consists of a string.

Output is a string Valid or Invalid.

Refer sample output for formatting specifications.

**Sample Input 1:**

ashok\_23

**Sample Output 1:**

Valid

**Sample Input 2:**

1980\_200

**Sample Output 2:**

Invalid

**27.Validating Input Password**

**import** java.util.Scanner;

**class** UserMainCode {

**public static int** validatePassword(String s1)

{

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

**if**(k>=8 && s1.matches(".\*[@\_#]{1}.\*") && s1.matches("^[^@\_#0-9].\*") && s1.matches(".\*[^@\_#]$")) // ^ mean not in the selected range

{

**return** 1;

}

**else**

**return** -1;

}

**Public static void** main(String[] args) {

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

String s1=in.nextLine();

**int** k=UserMainCode.*validatePassword*(s1);

**if**(k==1)

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

**else**

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

in.close();

}

}

**28.iD Validation**

Write a program to get two string inputs and validate the ID as per the specified format.

Include a class **UserMainCode** with a static method **validateIDLocations** which accepts two strings as input.

The return type of the output is a string Valid Id or Invalid Id.

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

**Input and Output Format:**

Input consists of two strings.

First string is ID and second string is location. ID is in the format CTS-LLL-XXXX where LLL is the first three letters of given location and XXXX is a four digit number.

Output is a string Valid id or Invalid id.

Refer sample output for formatting specifications.

**Sample Input 1:**

CTS-hyd-1234

hyderabad

**Sample Output 1:**

Valid id

**Sample Input 2:**

CTS-hyd-123

hyderabad

**Sample Output 2:**

Invalid id

**28.iD Validation**

**import**java.util.Scanner;

**publicclass**UserMainCode {

**publicstatic** String validateIDLocations(String s1,String s2)

{

**if**(s1.matches("(CTS)[-]{1}[a-zA-Z]{3}[-]{1}[0-9]{4}"))

{

**if**(s1.charAt(4)==s2.charAt(0) && s1.charAt(5)==s2.charAt(1) && s1.charAt(6)==s2.charAt(2))

**return**"valid";

}

**return**"invalid";

}

**Public static void** main(String[] args) {

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

String s1=in.nextLine();

String s2=in.nextLine();

System.*out*.println(UserMainCode.*validateIDLocations*(s1,s2));

in.close();

}

}

**29.Remove Elements**

Write a program to remove all the elements of the given length and return the size of the final array as output. If there is no element of the given length, return the size of the same array as output.

Include a class **UserMainCode** with a static method **removeElements** which accepts a string array, the number of elements in the array and an integer. The return type (integer) should return the size of the final array as output.

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

Assume maximum length of array is 20.

**Input and Output Format:**

Input consists of a integers that corresponds to n, followed by n strings and finally m which corresponds to the length value.

Output consists of a single Integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

5

a

bb

b

ccc

ddd

2

**Sample Output 1:**

4

**29.Remove Elements**

**import** java.util.Scanner;

**class** UserMainCode {

**public static int** removeElements(String s1[],**int** k)

{

**int** a=s1.length;

**for**(**int** i=0;i<s1.length;i++) // s.length for functions in char array

{

**if**(s1[i].length()==k) // s.length() for functions in string array

--a;

}

**return** a;

}

**public static void** main(String[] args) {

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

**int** l=Integer.*parseInt*(in.nextLine());

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

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

s1[i]=in.nextLine();

**int** k=Integer.*parseInt*(in.nextLine());

System.*out*.println(UserMainCode.*removeElements*(s1,k));

in.close();

}

}

**(or)**

**import**java.util.Scanner;

**publicclass**UserMainCode {

**publicstaticint**removeElements(String s1[],**int** k)

{

**int** a=s1.length;

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

{

**if**(s1[i].length()==k)

--a;

}

**return** a;

}

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

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

**int** l=in.nextInt();

in.nextLine();

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

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

s1[i]=in.nextLine();

**int** k=in.nextInt();

in.nextLine();

System.*out*.println(UserMainCode.*removeElements*(s1,k));

in.close();

}

}

**32.IP Validator**

Write a program to read a string and validate the IP address. Print “Valid” if the IP address is valid, else print “Invalid”.

Include a class **UserMainCode** with a static method **ipValidator** which accepts a string. The return type (integer) should return 1 if it is a valid IP address else return 2.

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 that corresponds to an IP.

Output consists of a string(“Valid” or “Invalid”).

Refer sample output for formatting specifications.

**Note**: An IP address has the format a.b.c.d where a,b,c,d are numbers between 0-255.

**Sample Input 1:**

132.145.184.210

**Sample Output 1:**

Valid

**Sample Input 2:**

132.145.184.290

**Sample Output 2:**

Invalid

**32.IP Validator**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** UserMainCode {

**public** **static** String ipValidator(String s1)

{

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

**int** i=0;

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

{

**int** k=Integer.*parseInt*(st.nextToken());

**if**(k>=0 && k<=255)

++i;

}

**if**(i==4)

**return**"VALID";

**return**"Invalid";

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*ipValidator*(s1));

in.close();

}

}

**34.File Extension**

Write a program to read a file name as a string and find out the file extension and return it as output. For example, the file sun.gif has the extension gif.

Include a class **UserMainCode** with a static method **fileIdentifier** which accepts a string. The return type (string) should return the extension of the input string (filename).

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 that corresponds to a file name.

Output consists of a string(extension of the input string (filename)).

Refer sample output for formatting specifications.

**Sample Input 1:**

sun.gif

**Sample Output 1:**

Gif

**34.File Extension**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**class** UserMainCode {

**public static** String ipValidator(String s1)

{

StringTokenizerst=**new**StringTokenizer(s1,".");

st.nextToken();

String s=st.nextToken();

**return** s;

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*ipValidator*(s1));

in.close();

}

}

**35.Find common characters and unique characters in string**

Given a method with two strings as input. Write code to count the common and unique letters in the two strings.

Note:

- Space should not be counted as a letter.

- Consider letters to be case sensitive. ie, "a" is not equal to "A".

Include a class **UserMainCode** with a static method **commonChars** which accepts two strings as input.

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

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

**Input and Output Format:**

Input consists of two strings.

Output is an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

a black cow

battle ship

**Sample Output 1:**

2  
  
[**Explanation** : b, l and a are the common letters between the 2 input strings. But 'a' appears more than once in the 1st string. So 'a' should not be considered while computing the count value.]

**Sample Input 2:**

australia

sri lanka

**Sample Output 2:**

4

**35.Find common characters and unique characters in string**

**import** java.util.Scanner;

**public** **class** UserMainCode

{

**static** **int** uniqueCounter(String s1,String s2)

{

**int** i,l=0,j;

String temp,temp1;

s1=s1.replaceAll("\\s","");

s2=s2.replaceAll("\\s","");

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

{

**if**(i!=0)

temp=s1.substring(0,i).concat(s1.substring(i+1));

**else**

temp=s1.substring(i+1);

String c=s1.charAt(i)+"";

**if**((!temp.contains(c)) && s2.contains(c))

{

**int** k=s2.indexOf(c);

**if**(k!=0)

temp1=s1.substring(0,k).concat(s1.substring(k+1));

**else**

temp1=s1.substring(k+1);

String d=s1.charAt(k)+"";

**if**(!temp.contains(c))

{

++l;

}

}

}

**return** l;

}

**public** **static** **void** main(String args[])

{

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

String str=in.nextLine();

String str1=in.nextLine();

**int** ans=UserMainCode.*uniqueCounter*(str,str1);

System.*out*.println(ans);

in.close();

}

}

**36.Initial Format**

Write a program to input a person's name in the format "FirstName LastName" and return the person name in the following format - "LastName, InitialOfFirstName".

Include a class **UserMainCode** with a static method **nameFormatter** which accepts a string. The return type (string) should return the expected format.

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 that corresponds to a Person's name.

Output consists of a string(person's name in expected format).

Refer sample output for formatting specifications.

**Sample Input :**

Jessica Miller

**Sample Output:**

Miller, J

**36.Initial Format**

**import**java.util.Scanner;

**import**java.util.StringTokenizer;

**publicclass**UserMainCode {

**publicstatic** String nameFormatter(String s1)

{

StringBuffersb=**new**StringBuffer();

StringTokenizerst=**new**StringTokenizer(s1);

String s2=st.nextToken();

String s3=st.nextToken();

sb.append(s3);

sb.append(",");

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

**return**sb.toString();

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*nameFormatter*(s1));

in.close();

}

}

**37.Character cleaning**

Write a program to input a String and a character, and remove that character from the given String. Print the final string.

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

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 and a character.

Output consists of a string(the character cleaned string).

Refer sample output for formatting specifications.

**Sample Input :**

elephant

e

**Sample Output:**

Lphant

**37.Character cleaning**

**import**java.util.Scanner;

**publicclass**UserMainCode {

/\*\*

\* **@param**args

\*/

**static**StringremoveCharacter(String s1,**char** s2)

{

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

StringBuffer sb1= **new**StringBuffer();

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

{

**char** c=s1.charAt(i);

**if**(c!=s2)

sb1.append(c);

}

**return** sb1.toString();

}

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

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

**char** n2=in.next().charAt(0);

String n3=UserMainCode.*removeCharacter*(n1,n2);

System.*out*.println(n3);

}

}

**38.Vowel Check**

Write a program to read a String and check if that String contains all the vowels. Print “yes” if the string contains all vowels else print “no”.

Include a class **UserMainCode** with a static method **getVowels** which accepts a string. The return type (integer) should return 1 if the String contains all vowels else return -1.

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

**Input and Output Format:**

Input consists of a string.

Output consists of a string(“yes” or “no”).

Refer sample output for formatting specifications.

**Sample Input 1:**

abceiduosp

**Sample Output 1:**

yes

**Sample Input 2:**

bceiduosp

**Sample Output 2:**

No

**38.Vowel Check**

**import**java.util.Scanner;

**publicclass**UserMainCode {

**publicstatic**StringgetVowels(String s1)

{ String s=”yes”;

**if**(s1.contains(“a”) && s1.contains(“e”) &&s1.contains(“i”) &&s1.contains(“o”) && s1.contains(“u”))

s=yes;

else

s=”no”;

**return** s;

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*getVowels*(s1));

in.close();

}

}

**39.Swap Characters**

Write a program to input a String and swap the every 2 characters in the string. If size is an odd number then keep the last letter as it is. Print the final swapped string.

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

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:**

TRAINER

**Sample Output 1:**

RTIAENR

**Sample Input 2:**

TOM ANDJERRY

**Sample output 2:**

OT MNAJDREYR

**39.Swap Characters**

**import** java.util.Scanner;

**public class** UserMainCode {

**public static** String swapCharacter(String s1)

{

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

**for**(**int** i=0;i<s1.length();i=i+2)

{ **char** c,d;

**if**(i!=s1.length()-1) //just to check that whether is it the last element

{

c=s1.charAt(i);

d=s1.charAt(i+1);

sb.append(d); //enter d first then c

sb.append(c);

}

**else**

{

c=s1.charAt(i);

sb.append(c);

}

}

**Return** sb.toString();

}

**Public static void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*swapCharacter*(s1));

in.close();

}

}

**42.Count Sequential Characters**

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

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

The return type of the output is the repeat count.

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

**Input and Output Format:**

Input consists a string.

Output is an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

abcXXXabc

**Sample Output 1:**

1

**Sample Input 2:**

aaaxxyzAAAx

**Sample Output 2:**

2

**42.Count Sequential Characters**

**import**java.util.Scanner;

**publicclass**UserMainCode {

**staticint**countSequentialChars(String str1)

{

**int** c=0;

**for**(**int**i=0;i<str1.length()-1;i++)

{

**if**(str1.charAt(i)==str1.charAt(i+1))

{

**if**(str1.charAt(i+1)==str1.charAt(i+2))

{

++c;

i=i+2;

}

}

}

**return** c;

}

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

{

String s1;

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

s1=in.nextLine();

**int**ans=UserMainCode.*countSequentialChars*(s1);

System.*out*.println(ans);

}

}

**43.Length of the Largest Chunk**

Write a program to read a string and find the length of the largest chunk in the string. If there are no chunk print “No chunks” else print the length.

NOTE: chunk is the letter which is repeating 2 or more than 2 times.

Include a class **UserMainCode** with a static method **largestChunk** which accepts a string. The return type (Integer) should return the length of the largest chunk if the chunk is present, else return -1.

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

**Input and Output Format:**

Input consists of a string.

Refer sample output for formatting specifications.

**Sample Input 1:**

You are toooo good

**Sample Output 1:**

4

**(Because the largest chunk is letter 'o' which is repeating 4 times)**

**Sample Input 2:**

who are u

**Sample Output 2:**

No chunks

**43) largest chunks**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public class** UserMainCode {

**static int** largestChunk(String str1)

{

**int** c,max=0;

StringTokenizerst=**new**StringTokenizer(str1);

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

{

String s1=st.nextToken();

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

{

c=0;

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

{

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

{

++c;

}

}

**if**(c>max)

max=c;

}

}

**return** (max);

}

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

{

String s1;

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

s1=in.nextLine();

**int**ans=UserMainCode.*largestChunk*(s1);

**if**(ans>=2)

System.*out*.println(ans);

**else**

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

}

}

**(or)**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public class** UserMainCode {

**static int** largestChunk(String str1)

{

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

StringTokenizer st=**new** StringTokenizer(str1);

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

{

String s1=st.nextToken();

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

{

c=1;

**if**(s1.charAt(i)==s1.charAt(i+1))

{ ++c;

**for**(j=i+2;j<s1.length();j++)

{

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

{

++c;

}

**else**

**break**;

}

}

**if**(c>max)

{

max=c;

i=j-1;

}

}

}

**return** (max);

}

**public static void** main(String args[])

{

String s1;

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

s1=in.nextLine();

**int**ans=UserMainCode.*largestChunk*(s1);

**if**(ans>=2)

System.*out*.println(ans);

**else**

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

}

}

**44.Unique Characters in a string**

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

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

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

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

**Input and Output Format:**

Input consists a string.

Output is an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

HelloWorld

**Sample Output 1:**

5

**Sample Input 2:**

coco

**Sample Output 2:**

-1

**44 .Unique Characters in a string**

**import** java.util.Scanner;

**public** **class** UserMainCode

{

**static** **int** uniqueCounter(String s)

{

**int** i,l=0;

String temp;

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

{

**if**(i!=0)

temp=s.substring(0,i).concat(s.substring(i+1));

**else**

temp=s.substring(i+1);

String c=s.charAt(i)+"";

**if**(!temp.contains(c))

{

++l;

}

}

**return** l;

}

**public** **static** **void** main(String args[])

{

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

String str=in.nextLine();

**int** ans=UserMainCode.*uniqueCounter*(str);

System.*out*.println(ans);

in.close();

}

}

**45.Name Shrinking**

Write a program that accepts a string as input and converts the first two names into dot-separated initials and printa the output.  
  
Input string format is 'fn mn ln'.   Output string format is 'ln [mn's 1st character].[fn's 1st character]'

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

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

**Input and Output Format:**

Input consists of a string.

Output consists of a String.

Refer sample output for formatting specifications.

**Sample Input:**

Sachin Ramesh Tendulkar

**Sample Output:**

Tendulkar R.S

**45.Name Shrinking**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** UserMainCode {

**public** **static** String getFormatedString(String s1)

{

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

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

String s2=st.nextToken();

String s3=st.nextToken();

String s4=st.nextToken();

sb.append(s4).append(" ");

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

sb.append(".");

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

**return** sb.toString();

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*getFormatedString*(s1));

in.close();

}

}

**46.Odd Digit Sum**

Write a program to input a String array. The input may contain digits and alphabets (“de5g4G7R”). Extract odd digits from each string and find the sum and print the output.

For example, if the string is "AKj375A" then take 3+7+5=15 and not as 375 as digit.

Include a class **UserMainCode** with a static method **oddDigitSum** which accepts a string array and the size of the array. The return type (Integer) should return the sum.

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

Assume maximum length of array is 20.

**Input and Output Format:**

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

Output consists of an Integer.

Refer sample output for formatting specifications.

**Sample Input :**

3

cog2nizant1

al33k

d2t4H3r5

**Sample Output :**

15

**(1+3+3+3+5)**

**46.Odd Digit Sum**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** **int** oddDigitSum(String s1[],**int** k)

{

**int** sum=0;

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

{

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

{

**char** c=s1[i].charAt(j);

**if**(c>='0' && c<='9')

{

**int** x=Integer.*parseInt*(c+"");

**if**(x%2!=0)

sum=sum+x;

}

}

}

**return** sum;

}

**public** **static** **void** main(String[] args) {

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

**int** l=Integer.*parseInt*(in.nextLine());

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

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

s1[i]=in.nextLine();

System.*out*.println(UserMainCode.*oddDigitSum*(s1,l));

in.close();

}

}

**49.Color Code Validation**

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

Validation Rule:

String should start with the Character '#'.

Length of String is 7.

It should contain 6 Characters after '#' Symbol.

It should contain Characters between 'A-F' and Digits '0-9'.

If String acceptable the return true otherwise false.

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

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

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

**Input and Output Format:**

Input consists a string corresponding to the color code.

Output is a boolean which returns true or false

Refer sample output for formatting specifications.

**Sample Input 1:**

#99FF33

**Sample Output 1:**

true

**Sample Input 2:**

#CCCC99#

**Sample Output 2:**

false

**49. Color Code Validation**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** String validateColorCode(String s1)

{

**if**(s1.matches("[#]{1}[A-F|0-9]{6}"))

**return** "true";

**else**

**return** "false";

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*validateColorCode*(s1));

in.close();

}

}

**50.Repeating set of characters in a string**

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

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

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

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

**Input and Output Format:**

Input consists a string and a positive integer n.

Output is a string with repeated characters.

Refer sample output for formatting specifications.

**Sample Input 1:**

Cognizant

3

**Sample Output 1:**

Cognizantantantant

**Sample Input 2:**

myacademy

2

**Sample Output 2:**

myacademymymy

**50.Repeating set of characters in a string**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** String getString (String s1,**int** k)

{

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

StringBuffer sb=**new** StringBuffer(s1);

String s2=s1.substring(x-k);

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

sb.append(s2);

**return** sb.toString();

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

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

System.*out*.println(UserMainCode.*getString* (s1,a));

in.close();

}

}

**54.Flush Characters**

Write a program to read a string from the user and remove all the alphabets and spaces from the String, and **only store special characters and digit** in the output String. Print the output string.

Include a class **UserMainCode** with a static method **getSpecialChar**which accepts a string. The return type (String) should return the character removed 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 an String (character removed string).

Refer sample output for formatting specifications.

**Sample Input :**

cogniz$#45Ant

**Sample Output :**

$#45

**54.Flush Characters**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** String getString (String s1)

{

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

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

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

{ **char** c=s1.charAt(i);

**if**((!Character.*isAlphabetic*(c)) && c!=’ ‘)

sb.append(c);

}

**return** sb.toString();

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*getString* (s1));

in.close();

}

}

**55.String Repetition**

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

If input2 is equal or greater than 3 then repeat the first three character of the String by given input2 times, separated by a space.

If input2 is 2 then repeat the first two character of String two times separated by a space,

If input2 is 1 then return the first character of the String.

Include a class UserMainCode with a static method **repeatString** which takes a string & integer and returns a string 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 and integer.

Output consists of a string.

Refer sample output for formatting specifications.

**Sample Input 1:**

COGNIZANT

4

**Sample Output 1:**

COG COG COG COG

**Sample Input 2:**

COGNIZANT

2

**Sample Output 2:**

CO CO

**55.String Repetition**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** String repeatString(String s1,**int** n)

{

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

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

{

**if**(n>=3)

sb.append(s1.substring(0,3)).append(" ");

**else** **if**(n==2)

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

**else** **if**(n==1)

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

}

**return** sb.toString();

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

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

System.*out*.println(UserMainCode.*repeatString*(s1,a));

in.close();

}

}

**59.Simple String Manipulation**

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

Return the String without the first 2 chars except when

1. keep the first char if it is 'j'
2. keep the second char if it is 'b'.

Include a class UserMainCode with a static method **getString** which accepts a string. The return type (string) should be the modified string based on the above rules. Consider all letters in the input to be small case.

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:**

hello

**Sample Output 1:**

llo

**Sample Input 2:**

java

**Sample Output 2:**

Jva

**59.Simple String Manipulation**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** String getString(String s1)

{

**char** c=s1.charAt(0);

**char** d=s1.charAt(1);

String s=s1.substring(2);

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

**if**(c!='j' && d!='b')

sb.append(s);

**else** **if**(c!='j' && d=='b')

sb.append(d).append(s);

**else** **if**(c=='j' && d!='b')

sb.append(c).append(s);

**else**

sb.append(s1);

**return** sb.toString();

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*getString*(s1));

in.close();

}

}

**62.Count Vowels**

Given a string input, write a program to find the total number of vowels in the given string.

Include a class **UserMainCode** with a static method “**countVowels**” that accepts a String argument and returns an int that corresponds to the total number of vowels in the given string.

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

**Input and Output Format:**

Input consists of a string.

Output consists of an integer..

**Sample Input:**

avinash

**Sample Output:**

3

**62.Count Vowels**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** **int** countVowels (String s1)

{

String s=s1.toLowerCase();

**int** c=0;

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

{

**char** k=s.charAt(i);

**if**(k=='a' || k=='e' || k=='i' || k=='o' || k=='u')

++c;

}

**return** c;

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode. countVowels (s1));

in.close();

}

}

**64.Reverse SubString**

Given a string, startIndex and length, write a program to extract the substring from right to left. Assume the last character has index 0.

Include a class **UserMainCode** with a static method “**reverseSubstring**” that accepts 3 arguments and returns a string. The 1st argument corresponds to the string, the second argument corresponds to the startIndex and the third argument corresponds to the length.

Create a class **Main** which would get a String and 2 integers as input and call the static method **reverseSubstring** present in the UserMainCode.

**Input and Output Format:**

The first line of the input consists of a string.

The second line of the input consists of an integer that corresponds to the startIndex.

The third line of the input consists of an integer that corresponds to the length of the substring.

**Sample Input:**

rajasthan

2

3

**Sample Output:**

hts

**64. Reverse SubString**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** String reverseSubstring(String s1,**int** n1,**int** n2)

{

StringBuffer sb=**new** StringBuffer(s1);

String s=sb.reverse().toString();

**return** s.substring(n1,n1+n2);

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

**int** n11=in.nextInt();

**int** n21=in.nextInt();

System.*out*.println(UserMainCode.*reverseSubstring*(s1,n11,n21));

in.close();

}

}

**65.String Finder**

Given three strings say Searchstring, Str1 and Str2 as input, write a program to find out if Str2 comes after Str1 in the Searchstring.

Include a class **UserMainCode** with a static method “**stringFinder**” that accepts 3 String arguments and returns an integer. The 3 arguments correspond to SearchString, Str1 and Str2. The function returns 1 if Str2 appears after Str1 in the Searchtring. Else it returns 2.

Create a class **Main** which would get 3 Strings as input and call the static method **stringFinder** present in the UserMainCode.

**Input and Output Format:**

Input consists of 3 strings.

The first input corresponds to the SearchString.

The second input corresponds to Str1.

The third input corresponds to Str2.

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

**Sample Input 1:**

geniousRajKumarDev

Raj

Dev

**Sample Output 1:**

yes

**Sample Input 2:**

geniousRajKumarDev

Dev

Raj

**Sample Output 2:**

No

**65.String Finder**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** **int** stringFinder(String s1,String s2,String s3)

{

**int** l=0;

**if**(s1.contains(s2)&& s1.contains(s3))

{

**if**(s1.indexOf(s2)<s1.lastindexOf(s3))

{

l=1;

}

**else**

l=2;

}

**return** l;

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

String s2=in.nextLine();

String s3=in.nextLine();

**int** ans=UserMainCode.*stringFinder*(s1,s2,s3);

**if**(ans==1)

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

**else**

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

in.close();

}

}

**66.Phone Number Validator**

Given a phone number as a string input, write a program to verify whether the phone number is valid using the following business rules:

-It should contain only numbers or dashes (-)

- dashes may appear at any position

-Should have exactly 10 digits

Include a class **UserMainCode** with a static method “**validatePhoneNumber**” that accepts a String input and returns a integer. The method returns 1 if the phone number is valid. Else it returns 2.

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

265-265-7777

**Sample Output 1:**

Valid

**Sample Input 2:**

265-65-7777

**Sample Output 1:**

Invalid

**66.Phone Number Validator**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** **int** validatePhoneNumber(String s1)

{

**int** l=0;

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

{

**char** c=s1.charAt(i);

**if**(Character.*isDigit*(c))

{

++l;

}

**else** **if**(c!='-')

**break**;

}

**if**(l==10)

**return** 1;

**else**

**return** 0;

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

**int** ans=UserMainCode.*validatePhoneNumber*(s1);

**if**(ans==1)

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

**else**

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

in.close();

}

}

**68.Negative String**

Given a string input, write a program to replace every appearance of the word "is" by "is not".

If the word "is" is immediately preceeded or followed by a letter no change should be made to the string .

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

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

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input 1:**

This is just a misconception

**Sample Output 1:**

This is not just a misconception

**Sample Input 2:**

Today is misty

**Sample Output 2:**

Today is not misty

**68. Negative String**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** UserMainCode {

**public** **static** String negativeString(String s1)

{

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

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

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

{

String s2=st.nextToken();

**if**(s2.equals("is"))

{

String s3=s2.replace("is", "is not");

sb.append(s3).append(" ");

}

**else**

{

sb.append(s2).append(" ");

}

}

**return** sb.toString();

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*negativeString*(s1));

in.close();

}

}

**69.Validate Number**

Given a negative number as string input, write a program to validate the number and to print the corresponding positive number.

Include a class **UserMainCode** with a static method “**validateNumber**” that accepts a string argument and returns a string. If the argument string contains a valid negative number, the method returns the corresponding positive number as a string. Else the method returns -1.

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

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input 1:**

-94923

**Sample Output 1:**

94923

**Sample Input 2:**

-6t

**Sample Output 2:**

-1

**69.Validate Number**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** String validateNumber(String s1)

{

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

String s="invalid";

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

{

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

{

**char** ch=s1.charAt(i);

**if**(ch>='0' && ch<='9')

++l;

**else**

**break**;

}

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

{

**int** d=Integer.*parseInt*(s1);

c=(-1)\*d;

s=String.*valueOf*(c);

}

**else**

s="invalid";

}

**return** s;

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*validateNumber*(s1));

in.close();

}

}

**71.String Processing - III**

Write a program to read a string where all the lowercase 'x' chars have been moved to the end of the string.  
  
Include a class UserMainCode with a static method **moveX** 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:**  
xxhixx  
  
**Sample Output 1:**  
hixxxx  
  
**Sample Input 2:**  
XXxxtest  
  
**Sample Output 2:**  
XXtestxx

**71.String Processing - III**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**static** String moveX(String s1)

{

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

StringBuffer sb1=**new** StringBuffer();

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

{

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

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

**else**

sb1.append(s1.charAt(i));

}

**return** sb1.append(sb).toString();

}

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

{

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

String str=in.nextLine();

System.*out*.println(UserMainCode.*moveX*(str));

in.close();

}

}

**72.String Processing - IV**

Write a program to read a string and also a number N. Form a new string starting with 1st character and with every Nth character of the given string. Ex - if N is 3, use chars 1, 3, 6, ... and so on to form the new String. Assume N>=1.  
  
Include a class UserMainCode with a static method **getStringUsingNthCharacter** which accepts the string and the number n. The return type is the string as per the problem statement.  
  
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:**  
HelloWorld  
2  
  
**Sample Output 1:**  
HelWrd

**72.String Processing - IV**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**static** String moveX(String s1,**int** a)

{

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

sb.append(s1.charAt(0));

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

{

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

}

**return** sb.toString();

}

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

{

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

String str=in.nextLine();

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

System.*out*.println(UserMainCode.*moveX*(str,n));

in.close();

}

}

**75.String Processing - MixMania**

Write a program to read a string and check if it starts with '\_ix' where '\_' is any one char(a-z, A-Z, 0-9).  
  
If specified pattern is found return true else false.  
  
Include a class UserMainCode with a static method **checkPattern** 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:**  
Mix Mania  
  
**Sample Output 1:**  
TRUE

**75.String Processing – MixMania**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**static**String checkPattern(String s1)

{

**if**(s1.matches("[a-zA-Z0-9]{1}(ix).\*"))

**return**"true";

**return**"false";

}

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

{

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

String str=in.nextLine();

System.*out*.println(UserMainCode.*checkPattern*(str));

in.close();

}

}

**76.String Processing**

Write a program to read a string and return a new string where the first and last chars have been interchanged.  
  
Include a class UserMainCode with a static method **exchangeCharacters** 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 string.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
HelloWorld  
  
**Sample Output 1:**  
delloWorlH

**76.String Processing**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**static** String checkPattern(String s1)

{

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

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

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

sb.append(s1.substring(1,x-1)).append(s1.charAt(0));

**return** sb.toString();

}

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

{

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

String str=in.nextLine();

System.*out*.println(UserMainCode.*checkPattern*(str));

in.close();

}

}

**77.Regular Expression - II**

Given a string (s) apply the following rules.  
  
1. String consists of three characters only.  
2. The characters should be alphabets only.  
  
If all the conditions are satisifed then print TRUE else print FALSE.  
  
Include a class UserMainCode with a static method **validateString** 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:**  
AcB  
  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
A2B  
  
**Sample Output 2:**  
FALSE

**77.Regular Expression - II**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**static** String validateString(String s1)

{

**if**(s1.matches("[a-zA-Z]{3}"))

**return**"true";

**else**

**return**"false";

}

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

{

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

String str=in.nextLine();

System.*out*.println(UserMainCode.*validateString*(str));

in.close();

}

}

**78.Strings Processing - Replication**

Write a program to read a string and also a number N. Return the replica of original string for n given time.  
  
Include a class UserMainCode with a static method **repeatString** which accepts the the string and the number n. The return type is the string based on the problem statement.  
  
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:**  
Lily  
2  
  
**Sample Output 1:**  
LilyLily

**78.Strings Processing - Replication**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**static**String validateString(String s1,**int** a)

{

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

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

{

sb.append(s1);

}

**return** sb.toString();

}

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

{

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

String str=in.nextLine();

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

System.*out*.println(UserMainCode.*validateString*(str,n));

in.close();

}

}

**80.String Processing - V**

Write a program to read a string array, concatenate the array elements one by one separated by comma and return the final string as output.  
Include a class UserMainCode with a static method **concatString** which accepts the string array. The return type is the string.  
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 an integer n which is the number of elements followed by n string values.  
Output consists of the string.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
3  
AAA  
BBB  
CCC  
**Sample Output 1:**  
AAA,BBB,CCC

**80.String Processing – V**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**static** String validateString(String s1[],**int** a)

{

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

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

{

sb.append(s1[i]).append(",");

}

sb.deleteCharAt(sb.length()-1);

**return** sb.toString();

}

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

{

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

**int** n=Integer.*parseInt*(in.nextLine());

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

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

str[i]=in.nextLine();

System.*out*.println(UserMainCode.*validateString*(str,n));

in.close();

}

}

**82.Math Calculator**

Write a program that accepts three inputs, first two inputs are operands in int form and third one being one of the following five operators: +, -, \*, /, %. Implement calculator logic and return the result of the given inputs as per the operator provided. In case of division, Assume the result would be integer.  
Include a class UserMainCode with a static method **calculator** which accepts two integers, one operand and returns the integer.  
Create a Class Main which would be used to accept three integers and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of two integers and a character.  
Output consists of a integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
23  
2  
\*  
**Sample Output 1:**  
46

**82.Math Calculator**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**staticint**calculator(**int** a,**int** b,**char** c)

{

**int** ans=0;

**switch**(c)

{

**case**'+':

ans=a+b;

**break**;

**case**'-':

ans=a-b;

**break**;

**case**'\*':

ans=a\*b;

**break**;

**case**'/':

ans=a/b;

**break**;

**case**'%':

ans=a%b;

**break**;

}

**return** ans;

}

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

{

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

**int** a=Integer.*parseInt*(in.nextLine());

**int** b=Integer.*parseInt*(in.nextLine());

**char** c=in.next().charAt(0);

System.*out*.println(UserMainCode.*calculator*(a,b,c));

in.close();

}

}

**85.Word Count**

Given a string array (s) and non negative integer (n) and return the number of elements in the array which have same number of characters as the givent int N.  
Include a class UserMainCode with a static method **countWord** which accepts the string array 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 indicating the number of elements in the string array followed the elements and ended by the non-negative integer (N).  
Output consists of a integer .  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
4  
a  
bb  
b  
ccc  
1  
**Sample Output 1:**  
2  
  
**Sample Input 2:**  
5  
dog  
cat  
monkey  
bear  
fox  
3  
**Sample Output 2:**  
3

**85.Word Count**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**staticint** WordCount(String s[],**int** x,**int** y)

{

**int** ans=0;

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

{

**if**(s[i].length()==y)

{

ans++;

}

}

**return** ans;

}

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

{

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

**int** n=Integer.*parseInt*(in.nextLine());

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

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

str[i]=in.nextLine();

**int** a=Integer.*parseInt*(in.nextLine());

System.*out*.println(UserMainCode.*WordCount*(str,n,a));

in.close();

}

}

**87.Word Count - II**

Write a program to read a string and count the number of words present in it.  
Include a class UserMainCode with a static method **countWord** which accepts the string. The return type is the integer giving out the count of words.  
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:**  
Today is Sunday  
**Sample Output 1:**  
3

**87.Word Count – II**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**publicclass** UserMainCode

{

**staticint** WordCount(String s)

{

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

**int** n=st.countTokens();

**return** n;

}

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

{

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

String str=in.nextLine();

System.*out*.println(UserMainCode.*WordCount*(str));

in.close();

}

}

**90.String Processing - V**

Write a program to read a string and also a number N. Form a new string made up of n repetitions of the last n characters of the String. You may assume that n is between 1 and the length of the string.  
  
Include a class UserMainCode with a static method **returnLastRepeatedCharacters** which accepts the string and the number n. The return type is the string as per the problem statement.  
  
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:**  
Hello  
2  
**Sample Output 1:**  
lolo  
  
**Sample Input 2:**  
Hello  
3  
**Sample Output 2:**  
llollollo

**90.String Processing – V**

**import** java.util.Scanner;

**publicclass** UserMainCode {

**publicstatic** String getString (String s1,**int** k)

{

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

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

String s2=s1.substring(x-k);

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

sb.append(s2);

**return** sb.toString();

}

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

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

String s1=in.nextLine();

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

System.*out*.println(UserMainCode.*getString* (s1,a));

in.close();

}

}

**91.Regular Expression - III**

Given a string (s) apply the following rules.  
1. String should not begin with a number.  
If the condition is satisifed then print TRUE else print FALSE.  
Include a class UserMainCode with a static method **validateString** 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:**  
ab2  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
72CAB  
**Sample Output 2:**  
FALSE

**91.Regular Expression – III**

**import** java.util.Scanner;

**publicclass** UserMainCode {

**publicstatic** String validateString(String s1)

{

**if**(s1.matches("[^0-9]{1}.\*"))

**return**"true";

**return**"false";

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*validateString*(s1));

in.close();

}

}

**92.String Processing - TrimCat**

Write a program to read a string and return a new string which is made of every alternate characters starting with the first character. For example NewYork will generate Nwok, and Samurai will generate Smri.  
  
Include a class UserMainCode with a static method getAlternateChars 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 string.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
Hello  
  
**Sample Output 1:**  
Hlo

**92.String Processing - TrimCat**

**import** java.util.Scanner;

**publicclass** UserMainCode {

**publicstatic**String getAlternateChars(String s1)

{

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

**for**(**int** i=0;i<s1.length();i=i+2)

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

**return** sb.toString();

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*getAlternateChars*(s1));

in.close();

}

}

**93.String Processing - Username**

Write a program to read a valid email id and extract the username.  
  
Note - user name is the string appearing before @ symbol.  
  
Include a class UserMainCode with a static method fetchUserName 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 string.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
[admin@xyz.com](mailto:admin@xyz.com)  
  
**Sample Output 1:**  
admin

**93.String Processing - Username**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**publicclass** UserMainCode {

**publicstatic** String getAlternateChars(String s1)

{

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

**return** st.nextToken();

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*getAlternateChars*(s1));

in.close();

}

}

**94.String Processing - VII**

Write a program to read a two strings and one int value(N). check if Nth character of first String from start and Nth character of second String from end are same or not. If both are same return true else return false.  
Check need not be Case sensitive  
Include a class UserMainCode with a static method **isEqual** which accepts the two strings and a integer n. The return type is the TRUE / FALSE.  
Create a Class Main which would be used to read the strings and integer and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of two strings and an integer.  
Output consists of TRUE / FALSE .  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
AAAA  
abab  
2  
  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
MNOP  
QRST  
3  
  
**Sample Output 2:**  
FALSE

**94.String Processing - VII**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**publicstatic**String getAlternateChars(String s1,String s2,**int** n)

{

String c=s1.charAt(n-1)+"";

String d=s2.charAt(s2.length()-n)+"";

**if**(c.equalsIgnoreCase(d))

**return**"true";

**return**"false";

}

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

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

String s1=in.nextLine();

String s2=in.nextLine();

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

System.*out*.println(UserMainCode.*getAlternateChars*(s1,s2,n));

in.close();

}

}

**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!

**1.Start Case**

**publicclass** UserMainCode {

**publicstatic** String printCapitalized(String s1)

{

StringBuffer s5=**new** StringBuffer();

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

**while**(t.hasMoreTokens()){

String s2=t.nextToken();

String s3=s2.substring(0,1);

String s4=s2.substring(1, s2.length());

s5.append(s3.toUpperCase()).append(s4).append(" "); }

**return** s5.toString();

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*printCapitalized*(s1));

in.close();

}

}

**4.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

**4.PAN Card**

**import** java.util.Scanner;

**publicclass** UserMainCode {

**publicstatic** String validatePAN(String s1)

{

**if**(s1.matches("[A-Z]{3}[0-9]{4}[A-Z]{1}"))

**return**"valid";

**else**

**return**"invalid";

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*validatePAN*(s1));

in.close();

}

}

**6.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

**6.Test Vowels**

**import** java.util.Scanner;

**public class**UserMainCode {

**public static** String getVowels(String s1)

{ String s=”yes”;

**if**(s1.contains(“a”) && s1.contains(“e”) &&s1.contains(“i”) &&s1.contains(“o”) && s1.contains(“u”))

s=yes;

else

s=”no”;

**return** s;

}

**public cstatic void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*getVowels*(s1));

in.close();

}

}

**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

**7.Dash Check**

**import** java.util.Scanner;

**publicclass** UserMainCode {

**publicstatic** String compareDashes(String s1,String s2)

{

**int** i;

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

{

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

{

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

**break**;

}

}

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

**return**"valid";

**return**"invalid";

}

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

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

String s1=in.nextLine();

String s2=in.nextLine();

System.*out*.println(UserMainCode.*compareDashes*(s1,s2));

in.close();

}

}

**8.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

**8.Reverse Split**

**public class** UserMainCode {

**public static** String reshape(String s1,**char** c)

{

**int** i;

StringBuffer sb1=**new** StringBuffer();

**for**(i=s1.length()-1;i>=0;i--)

{

sb1.append(s1.charAt(i)).append(c);

}

sb1.deleteCharAt(sb1.length()-1);

**return** sb1.toString();

}

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

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

String s1=in.nextLine();

**char** c=in.next().charAt(0);

System.*out*.println(UserMainCode.*reshape*(s1,c));

in.close();

}

}

**10.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

**10.Last Letters**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**publicclass** UserMainCode {

**publicstatic** String getLastLetter(String s1)

{

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

StringBuffer sb1=**new** StringBuffer();

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

String str=st.nextToken();

sb1.append(str.substring(str.length()-1).toUpperCase());

sb1.append("$");

}

sb1.deleteCharAt(sb1.length()-1);

**return** sb1.toString();

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*getLastLetter*(s1));

in.close();

}

}

**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:**

**12.All Numbers**

**import** java.util.Scanner;

**publicclass** UserMainCode {

**publicstatic** String validateNumber(String s1[],**int** n)

{

**int** l=0;

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

{

**if**(s1[i].matches("[0-9.]\*"))

++l;

}

**if**(l==n)

**return**"valid";

**return**"invalid";

}

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

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

**int** n=Integer.*parseInt*(in.nextLine());

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

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

str[i]=in.nextLine();

System.*out*.println(UserMainCode.*validateNumber*(str,n));

in.close();

}

}

**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\

**14.Max Substring**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**publicclass** UserMainCode {

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

{

String s4="";

String s=s2+" ";

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

**int** max=0;

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

{

String s3=st.nextToken();

**int** n=s3.length();

**if**(n>max)

{

max=n;

s4=s3;

}

}

**return** s4;

}

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

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

String str=in.nextLine();

**char** str1=in.next().charAt(0);

System.*out*.println(UserMainCode.*extractMax*(str,str1));

in.close();

}

}

**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

**16.Simple String Manipulation – II**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**static** **int** validateNumber(String s1)

{

**int** a=s1.indexOf(" ");

**int** b=s1.lastIndexOf(" ");

String s=s1.substring(0,a);

String s2=s1.substring(b+1);

**if**(s.equals(s2))

**return** s.length();

**return** (s.length()+s2.length());

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

System.*out*.println(UserMainCode.*validateNumber*(n1));

}

}

**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

**23.Convert Format**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**publicclass** UserMainCode {

**publicstatic** String convertFormat(String s)

{

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

String s1=t.nextToken();

String s2=t.nextToken();

String s3=t.nextToken();

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

sb.append(s1.substring(0, s1.length()-1)).append('-');

sb.append(s1.charAt(s1.length()-1)).append(s2.charAt(0)).append('-');

sb.append(s2.substring(1, s2.length())).append(s3.charAt(0)).append('-');

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

**return** sb.toString();

}

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

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

String str=in.nextLine();

System.*out*.println(UserMainCode.*convertFormat*(str));

in.close();

}

}

**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

**29.String Occurances – II**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**static** **int** validateNumber(String s1,String n)

{

String s=s1.replace(n,"$");

**int** l=0;

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

**if**(s.charAt(i)=='$')

++l;

**return** l;

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

String n=in.nextLine();

System.*out*.println(UserMainCode.*validateNumber*(n1,n));

}

}

**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

**32.Repeat Front**

**import** java.util.Scanner;

**publicclass** UserMainCode {

**publicstatic** String convertFormat(String s,**int** n)

{

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

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

{

**if**(s.length()<3)

sb.append(s);

**else**

sb.append(s.substring(0,3));

}

**return** sb.toString();

}

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

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

String str=in.nextLine();

**int** n=Integer.*parseInt*(in.nextLine());

System.*out*.println(UserMainCode.*convertFormat*(str,n));

in.close();

}

}

**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

**34) Pattern Matcher**

**import** java.util.Scanner;

**publicclass** UserMainCode {

**staticint** validateNumber(String s1)

{

**if**(s1.matches("(CPT-)[0-9]{6}"))

{

**return** 1;

}

**else**

**return** -1;

}

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

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

**int** c=UserMainCode.*validateNumber*(n1);

**if**(c==1)

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

**else**

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

}

}

**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$

**35.Playing with String – I**

**import** java.util.Scanner;

**publicclass** UserMainCode {

**publicstatic** String stringFinder(String s1[],**int** n,**int** a)

{

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

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

{

**if**(s1[i].length()>=a)

sb.append(s1[i].substring(a-1,a));

**else**

sb.append("$");

}

**return** sb.toString();

}

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

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

**int** n=Integer.*parseInt*(in.nextLine());

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

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

s1[i]=in.nextLine();

**int** a=Integer.*parseInt*(in.nextLine());

System.*out*.println(UserMainCode.*stringFinder*(s1,n,a));

in.close();

}

}

**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

**36.Regular Expression – 1**

**import** java.util.Scanner;

**publicclass** UserMainCode

{

**static** String checkPattern(String s1)

{

**if**(s1.matches("[a-zA-Z0-9]{1}[R]{1}[0-9]{1}.\*") && s1.length()==4)

**return**"true";

**return** "false";

}

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

{

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

String str=in.nextLine();

System.*out*.println(UserMainCode.*checkPattern*(str));

in.close();

}

}

**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

AcB/TRUE

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

**import** java.util.Scanner;

**public** **class** UserMainCode {

**static** String validateNumber(String s1)

{

**if**(s1.matches("[0-9]\*"))

{

**int** a=Integer.*parseInt*(s1);

**if**(a>=21 && a<=25)

**return** "valid";

}

**return** "invalid";

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

System.*out*.println(UserMainCode.*validateNumber*(n1));

}

}

**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

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

**import** java.util.Scanner;

**public** **class** UserMainCode {

**static** String validateNumber(String s1)

{

**if**(s1.matches("[1-9]{2}[0-9]{8}"))

{

**return** "valid";

}

**return** "invalid";

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

System.*out*.println(UserMainCode.*validateNumber*(n1));

}

}

**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

**39.String Splitter**

**public** **class** UserMainCode {

**public** **static** String[] compareDashes(String s1,**char** c)

{

String s=c+" ";

StringTokenizer a=**new** StringTokenizer(s1,s);

**int** k=0;

String ans[]=**new** String[a.countTokens()];

**while**(a.hasMoreTokens())

{

String b=a.nextToken();

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

sb.reverse();

ans[k]=sb.toString().toLowerCase();

++k;

}

**return** ans;

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

**char** c=in.next().charAt(0);

String o[]=UserMainCode.*compareDashes*(s1,c);

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

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

in.close();

}

}

**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

**40. Vowel Count**

**import**java.util.Scanner;

**publicclass**UserMainCode {

**publicstaticint**countVowels(String s1)

{

String s=s1.toLowerCase();

**int** c=0;

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

{

**char** k=s.charAt(i);

**if**(k=='a' || k=='e' || k=='i' || k=='o' || k=='u')

++c;

}

**return** c;

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.countVowels(s1));

in.close();

}

}

**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

**41. Playing with String – II**

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** String[] stringFinder(String s1[])

{

String s2[]=**new** String[s1.length];

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

{

s2[i]=s1[i].toLowerCase();

}

Arrays.*sort*(s2);

**return** s2;

}

**public** **static** **void** main(String[] args) {

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

**int** n=Integer.*parseInt*(in.nextLine());

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

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

s1[i]=in.nextLine();

String o[]=UserMainCode.*stringFinder*(s1);

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

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

in.close();

}

}

**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

**44.Asterisk & Characters**

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** UserMainCode {

**public** **static** **boolean** scanStarNeighbors(String s1)

{

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

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

{

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

{

**int** n=i;

**char** c=s1.charAt(n-1);

**char** d=s1.charAt(n+1);

**if**(c==d)

b=**true**;

**else**

{

b=**false**;

**break**;

}

}

}

**return** b;

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*scanStarNeighbors*(s1));

in.close();

}

}

**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

**45.Occurance Count**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** UserMainCode {

**public** **static** **int** compareDashes(String s1,String s2)

{

**int** ans=0;

StringTokenizer t=**new** StringTokenizer(s1);

**while**(t.hasMoreTokens())

{

String s3=t.nextToken();

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

ans++;

}

**return** ans;

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

String s2=in.nextLine();

System.*out*.println(UserMainCode.*compareDashes*(s1,s2));

in.close();

}

}

**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

**46.Regular Expressions - III**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**static** **int** searchString(String s1,String n)

{

String s2=s1.toLowerCase();

String s3=n.toLowerCase();

String s=s2.replace(s3,"$");

**int** l=0;

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

**if**(s.charAt(i)=='$')

++l;

**return** l;

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

String n=in.nextLine();

System.*out*.println(UserMainCode.*searchString*(n1,n));

}

}

**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

**47. Strings Processing**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** UserMainCode {

**public** **static** String findFruitName(String s1,**int** n)

{

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

String s="";

**int** k=st.countTokens();

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

{

**if**(i==n)

{

s=st.nextToken();

**break**;

}

**else**

{

s=st.nextToken();

}

}

**return** s;

}

**public** **static** **void** main(String[] args) {

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

String s1=in.nextLine();

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

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

in.close();

}

}

**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

**48. Proper Case**

**publicclass** UserMainCode {

**publicstatic** String printCapitalized(String s1)

{

StringBuffer s5=**new** StringBuffer();

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

**while**(t.hasMoreTokens()){

String s2=t.nextToken();

String s3=s2.substring(0,1);

String s4=s2.substring(1, s2.length());

s5.append(s3.toUpperCase()).append(s4).append(" "); }

**return** s5.toString();

}

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

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

String s1=in.nextLine();

System.*out*.println(UserMainCode.*printCapitalized*(s1));

in.close();

}

}

**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

**49. Length of same word**

**import** java.util.Scanner;

**public** **class** UserMainCode {

**static** **int** validateNumber(String s1)

{

**int** a=s1.indexOf(" ");

**int** b=s1.lastIndexOf(" ");

String s=s1.substring(0,a);

String s2=s1.substring(b+1);

**if**(s.equals(s2))

**return** s.length();

**return** (s.length()+s2.length());

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

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

String n1=in.nextLine();

System.*out*.println(UserMainCode.*validateNumber*(n1));

}

}

**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

**51. Find Digits**

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** UserMainCode {

**static** String validateNumber(**double** d)

{

**int** n1=0;

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

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

String s1=t.nextToken();

String s2=t.nextToken();

n1=s1.length();

**int** n3=0;

**for**(**int** i=s2.length()-1;i>=0;i--)

{

**if**(s2.charAt(i)==0)

++n3;

**else**

**break**;

}

**if**(n2!=1)

n2=n2-n3;

**else**

n2=1;

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

**return** s3;

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

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

**double** d=in.nextDouble();

System.*out*.println(UserMainCode.*validateNumber*(d));

}

}

**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

**63.Largest Chunk**

**import**java.util.Scanner;

**import**java.util.StringTokenizer;

**publicclass**UserMainCode {

**staticint**largestChunk(String str1)

{

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

StringTokenizerst=**new**StringTokenizer(str1);

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

{

String s1=st.nextToken();

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

{

c=1;

**if**(s1.charAt(i)==s1.charAt(i+1))

{ ++c;

**for**(j=i+2;j<s1.length();j++)

{

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

{

++c;

}

**else**

**break**;

}

}

**if**(c>max)

{

max=c;

i=j-1;

}

}

}

**return** (max);

}

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

{

String s1;

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

s1=in.nextLine();

**int**ans=UserMainCode.*largestChunk*(s1);

**if**(ans>=2)

System.*out*.println(ans);

**else**

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

}

}

**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

**66.Regular Expression - III**  
**import** java.util.Scanner;

**public** **class** UserMainCode

{

**static** **int** passwordValidation(String s)

{

**if**(s.length()>=8&&s.matches(".\*[A-Z].\*")&&s.matches(".\*[a-z].\*")&&s.matches(".\*[0-9].\*")&&s.matches(".\*[^0-9A-Za-z].\*"))

**return** 1;

**return** -1;

}

**public** **static** **void** main(String[] args)

{

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

String s=in.nextLine();

**int** b=UserMainCode.*passwordValidation*(s);

**if**(b==1)

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

**else**

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

}

}

**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

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

n

**import** java.util.Scanner;

**public** **class** UserMainCode

{

**static** String passwordValidation(String s1,String s2)

{

**if**(s1.length()>s2.length())

**return** s1+s2+s1;

**return** s2+s1+s2;

}

**public** **static** **void** main(String[] args)

{

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

String s1=in.nextLine();

String s2=in.nextLine();

System.*out*.println(UserMainCode.*passwordValidation*(s1,s2));

}

}