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

```
public class Main {

public static void main(String[] args) {

String s1="zzzz";
```

```
System.out.println(stringFormatting(s1));

}

public static String stringFormatting(String s1) {

StringBuffer sb=new StringBuffer();

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

char c=s1.charAt(i);

if(i%2==0){

if(c==122)

c=(char) (c-25);

else{

c=(char) (c+1);}

sb.append(c);}

else

sb.append(c);}

return sb.toString();

}

}
```

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

```
public class UserMainCode {

        public static int display(String password){

          if(password.matches(".*[0-9]{1,}.*") && password.matches(".*[@#$]{1,}.*")
&& password.length()>=6 && password.length()<=20)

            {

                return 1;
```

```
        }
        else
        {
            return -1;
        }
    }

}
```

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

```
public class Main {
	public static void main(String[] args) {
		String s1="capacity";
		System.out.println(removeEvenElements(s1));
	}
	public static String removeEvenElements(String s1) {
		StringBuffer sb1=new StringBuffer();
		for(int i=0;i<s1.length();i++)
			if((i%2)==0)
				sb1.append(s1.charAt(i));
			else if((i%2)!=0)
				if(s1.charAt(i)!='a' && s1.charAt(i)!='e' && s1.charAt(i)!='i' &&
s1.charAt(i)!='o' && s1.charAt(i)!='u')
					if(s1.charAt(i)!='A' && s1.charAt(i)!='E' &&
s1.charAt(i)!='I' && s1.charAt(i)!='O' && s1.charAt(i)!='U')
					 sb1.append(s1.charAt(i));
		return sb1.toString();
	}
}
```

14.Sum of Powers of elements in an array

Given a method with an int array. Write code to find the power of each individual element accoding to its position index, add them up and return as output.

Include a class **UserMainCode** with a static method **getSumOfPower** which accepts an integer array as input.

The return type of the output is an integer which is the sum powers of each element in the array.

Create a **Main** class which gets integer array as an input and call the static method **getSumOfPower** present in the**UserMainCode.**

Input and Output Format:

Input is an integer array.First element corresponds to the number(n) of elements in an array.The next inputs corresponds to each element in an array.

Output is an integer .

Sample Input 1:

4
3
6
2
1

Sample Output 1:
12

Sample Input 2:

4
5
3
7
2

Sample Output 2:
61

```
public class useerm{

public static int display(int n,int[]a)
```

```
{

        {
                int sum=0;
                for(int i=0;i<n;i++)
                        sum=(int)(sum+Math.pow(a[i], i));
                return sum;
    }}}
```

15.Difference between largest and smallest elements in an array

Given a method taking an int array having size more than or equal to 1 as input. Write code to return the difference between the largest and smallest elements in the array. If there is only one element in the array return the same element as output.

Include a class **UserMainCode** with a static method **getBigDiff** which accepts a integer array as input.

The return type of the output is an integer which is the difference between the largest and smallest elements in the array.

Create a **Main** class which gets integer array as an input and call the static method **getBigDiff** present in the**UserMainCode.**

Input and Output Format:

Input is an integer array.First element in the input represents the number of elements in an array.

Size of the array must be >=1

Output is an integer which is the difference between the largest and smallest element in an array.

Sample Input 1:

4
3
6
2
1

Sample Output 1:
5

Sample Input 2:
4
5
3
7
2

Sample Output 2:
5

```
import java.util.Arrays;

public class kape1 {

public static int display(int []array)



{

	Arrays.sort(array);

	int n=array[array.length-1]-array[0];

	int b=array.length;

	if(b==1)

	{

		n=array[0];

	}
```

```
        return n;

        }

}
```

16.Find the element position in a reversed string array

Given a method with an array of strings and one string variable as input. Write code to sort the given array in reverse alphabetical order and return the postion of the given string in the array.

Include a class **UserMainCode** with a static method **getElementPosition** which accepts an array of strings and a string variable as input.

The return type of the output is an integer which is the position of given string value from the array.

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

Input and Output Format:

Input is an string array. First element in the input represents the size the array

Assume the position of first element is 1.

Output is an integer which is the position of the string variable

Sample Input 1:
4
red
green
blue
ivory
ivory

Sample Output 1:
2

Sample Input 2:

3
grape
mango
apple

apple

Sample Output 2:
3

17.Generate the series

Given a method taking an odd positive Integer number as input. Write code to evaluate the following series:
1+3-5+7-9…+/-n.

Include a class **UserMainCode** with a static method **addSeries** which accepts a positive integer .

The return type of the output should be an integer .

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

Input and Output Format:

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

Sample Input 1:

9

Sample Output 1:

-3

Sample Input 2:

11

Sample Output 2:

8

```
public class UserMainCode
{
	public static int generateSeries(int n)
	{
		int i=0,sumo=0,sume=0,sum=1;
		if(n==1)
		{
			sum=n;
```

```
                    break;
                }


for(i=3;i<=n;i=i+4)
{


                sumo=sumo+i;
}
for(i=5;i<=n;i=i+4)
{
        sume=sume+i;
}



sum+=sumo-sume;
return sum;
        }
        }
```

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

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


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s1=sc.nextLine();
String s2=sc.nextLine();
int c=sc.nextInt();
System.out.println(UserMainCode.calculateElectricityBill(s1,s2, c));



}
}



public class UserMainCode {
	public static int calculateElectricityBill(String s1,String s2,int c)

	{

		int a=Integer.parseInt(s1.substring(5));
		int b=Integer.parseInt(s2.substring(5));
```

```
            int res=Math.abs((b-a)*c);
            return res;
        }
}
```

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

```
public class Main {

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

```
String s1="goodbad";
getvalues(s1);
}
public static void getvalues(String s1) {
int sum=0;
for(int i=0;i<s1.length();i++)
{
char a=s1.charAt(i);
if(Character.isDigit(a))
{
int b=Integer.parseInt(String.valueOf(a));
sum=sum+b;
}
}
if(sum==0)
{
System.out.println(-1);
}
else
System.out.println(sum);
}
}
```

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

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

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

```
Scanner sc=new Scanner(System.in);
String s1=sc.nextLine();
String s2=sc.nextLine();

System.out.println(UserMainCode.concatString(s1,s2));



}
}



public class UserMainCode
{
	public static String concatString(String s1,String s2)
	{
	StringBuffer sb=new StringBuffer();
	int n=s2.length();
	sb.append(s1.substring(s1.length()-n));
	sb.append(s2.substring(0));
	return sb.toString();
	}
}
```