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

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)
    {
    Scanner sc=new Scanner(System.in);
    String s=sc.nextLine();
```

```
        //UserMainCode u=new UserMainCode();
        int b=UserMainCode.validateColorCode(s);
        if(b==1)
        {
                System.out.println("Valid color code");

        }
        else
        {
                System.out.println("Invalid color code ");
        }

        }

}
public class UserMainCode
{
public static int validateColorCode(String a)
{
        int r=-1;
        if(a.matches("(#)[A-F0-9]{6}"))

        {
                r=1;
        }
        return r;
}
}
```

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.

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

****

CTS-215

```
public class Main {

public static void main(String[] args) {

String s1="CTS-2j4";

getvalues(s1);

}

public static void getvalues(String s1) {

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

{

System.out.println(1);

}

else
```

```
System.out.println(-1);

}

}
```

23.Removing Keys from HashMap

Given a method with a HashMap<Integer,string> as input. Write code to remove all the entries having keys multiple of 4 and return the size of the final hashmap.

Include a class **UserMainCode** with a static method **sizeOfResultandHashMap** which accepts hashmap as input.

The return type of the output is an integer which is the size of the resultant hashmap.

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

Input and Output Format:

First input corresponds to the size of the hashmap.

Input consists of a hashmap<integer,string>.

Output is an integer which is the size of the hashmap.

Refer sample output for formatting specifications.

Sample Input 1:

```
3
2
hi
4
hello
12
hello world
```

Sample Output 1:

```
1
```

Sample Input 2:

```
3
2
```

hi

4

sdfsdf

3

asdf

Sample Output 2:

2

24.Largest Element

Write a program to read an int array of odd length, compare the first, middle and the last elements in the array and return the largest. If there is only one element in the array return the same element.

Include a class **UserMainCode** with a static method **checkLargestAmongCorner** which accepts an int arrayThe return type (integer) should return the largest element among the first, middle and the last elements.

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

Assume maximum length of array is 20.

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Sample Input 1:

5

2

3

8

4

5

Sample Output 1:

8

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


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
int n=sc.nextInt();

int a[]=new int[n];
for(int i=0;i<n;i++)
{
	a[i]=sc.nextInt();
}
System.out.println(UserMainCode.checkLargestAmongCorner(a));



}
}
```

```
public class UserMainCode
{
	public static int checkLargestAmongCorner(int a[])
	{
		int max=0;
	int m=a[a.length/2];
	int f=a[0];
	int l=a[a.length-1];
	if(m>f && m>l)
	{
	  max=m;
	}
	else if(f>m && f>l)
	{
	  max=f;
	}
	else
	  max=l;

	return max;
	}
	}
```

25.nCr

Write a program to calculate the ways in which r elements can be selected from n population, using nCr formula nCr=n!/r!(n-r)! where first input being n and second input being r.

Note1 : n! factorial can be achieved using given formula n!=nx(n-1)x(n-2)x..3x2x1.
Note2 : 0! = 1.
Example 5!=5x4x3x2x1=120

Include a class **UserMainCode** with a static method **calculateNcr** which accepts two integers. The return type (integer) should return the value of nCr.

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

Input and Output Format:

Input consists of 2 integers. The first integer corresponds to n, the second integer corresponds to r.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Sample Input 1:

4
3

Sample Output 1:

4

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


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int r=sc.nextInt();

System.out.println(UserMainCode.calculateNcr(n,r) );



}
}



public class UserMainCode
{
	public static int calculateNcr(int n,int r)
	{
		int i,prod=1,prod1=1,prod2=1;
	for(i=1;i<=n;i++)
	{
```

```
        prod=prod*i;

    }
        for(i=1;i<=r;i++)
        {
            prod1=prod1*i;
        }
        int diff=n-r;
        for(i=1;i<=diff;i++)
        {
            prod2=prod2*i;
        }
        int dem=prod1*prod2;
        int res=prod/dem;
        return res;
    }
    }
```

26.Sum of Common Elements

Write a program to find out sum of common elements in given two arrays. If no common elements are found print - “No common elements”.
Include a class **UserMainCode** with a static method **getSumOfIntersection** which accepts two integer arrays and their sizes. The return type (integer) should return the sum of common elements.

Create a Class Main which would be used to accept 2 Input arrays and call the static method present in UserMainCode.

Input and Output Format:

Input consists of 2+m+n integers. The first integer corresponds to m (Size of the 1st array), the second integer corresponds to n (Size of the 2nd array), followed by m+n integers corresponding to the array elements.

Output consists of a single Integer corresponds to the sum of common elements or a string “No common elements”.

Refer sample output for formatting specifications.

Assume the common element appears only once in each array.

Sample Input 1:

4
3
2

3
5
1
1
3
9

Sample Output 1:

4

Sample Input 2:

4
3
2
3
5
1
12
31
9

Sample Output 2:

No common elements

```
public class Main {

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int m=sc.nextInt();

int[] a=new int[n];

int[] b=new int[m];
```

```
for(int i=0;i<n;i++)
a[i]=sc.nextInt();
for(int i=0;i<m;i++)
b[i]=sc.nextInt();
int u=UserMainCode.display(a,b);
if(u==-1)
System.out.println("No common elements");
else
System.out.println(u);}}


public class UserMainCode {
public static int display(int a[],int b[])
{


int sum=0;
for(int i=0;i<a.length;i++)
{
for(int j=0;j<b.length;j++)
{if(a[i]==b[j])
sum=sum+a[i];
}}
if(sum==0)
```

return -1;

else

return sum;

}}

27.Validating Input Password

102.Write a code get a password as string input 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

```
public class UserMainCode {
    public static int validatePassword(String password){
        String regEx = "[^0-9|@|_|#](.)*[@|_|#](.)*[^@|_|#]";
        if(password.length()>=8 && password.matches(regEx)){
            return 1;
        }
        return -1;
    }
}
```

```
if(s.length()>=8&&s.matches("[^0-9|#|_|@](.*)[#|_|@](.*)[^@|_|#]"))
{
      System.out.println("valid");
}
```

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

```
import java.util.*;

public class Main {

public static void main(String[] args) {

String s1="CTS-hyd-1234";

String s2="hyderabad";

boolean b=formattingString(s1,s2);

if(b==true)

System.out.println("String format:CTS-LLL-XXXX// valid id");

else

System.out.println("not in required format");

}

public static boolean formattingString(String s1, String s2) {

String s3=s2.substring(0, 3);
```

```
boolean b=false;

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

String s4=t.nextToken();

String s5=t.nextToken();

String s6=t.nextToken();

if(s4.equals("CTS") && s5.equals(s3) && s6.matches("[0-9]{4}"))

b=true;

else{

b=false;}

return b;

}

}
```

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

import java.util.*;

public class Main

{

public static void main(String[] args)

{

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

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

String[] a=new String[n];
for(int i=0;i<n;i++)
 a[i]=sc.nextLine();

int m=sc.nextInt();
```

System.out.println(UserMainCode.display(a,m));

}}

import java.util.*;

```
public class UserMainCode

{


        public static int display(String[] a,int m){


                        int u=a.length;

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

                        {

                        if(a[i].length()==m)

                        u--;

                        }

                        return u;

                        }}
```

30.Find the difference between Dates in months

Given a method with two date strings in yyyy-mm-dd format as input. Write code to find the difference between two dates in months.

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

The return type of the output is an integer which returns the diffenece between two dates in months.

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

Input and Output Format:

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

Sample Input 1:

2012-03-01

2012-04-16

Sample Output 1:

1

Sample Input 2:

2011-03-01
2012-04-16

Sample Output 2:

13

```
import java.text.*;

import java.util.*;

public class Main {

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

String s1="2012-03-01";

String s2="2012-03-16";

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

}

public static int monthsBetweenDates(String s1, String s2) throws ParseException {

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

```
Date d1=sdf.parse(s1);

Date d2=sdf.parse(s2);

Calendar cal=Calendar.getInstance();


cal.setTime(d1);

int months1=cal.get(Calendar.MONTH);

int year1=cal.get(Calendar.YEAR);

cal.setTime(d2);

int months2=cal.get(Calendar.MONTH);

int year2=cal.get(Calendar.YEAR);

int n=((year2-year1)*12)+(months2-months1);

return n;

}

}
```