IRC_JAVA_COD_REGEX

Test Summary

- No. of Sections: 1
- No. of Questions: 10
- Total Duration: 180 min

Section 1 - CODING

Section Summary

- No. of Questions: 10
- Duration: 180 min

Additional Instructions:

None

Q1. **Problem statement:**
Write a java program to find whether the password is valid or invalid using the regular expression.

Note:

1. Password should be less than or equal to 15 and more than 8 characters in length.
2. Password should contain at least one upper case and one lower case alphabet.
3. Password should contain at least one number.
4. Password should contain at least one special character.

Input Format

The input consists of a string that is in password form.

Output Format

The output displays whether the password is valid or invalid.

Sample Input

```
Iamneo@1
```

Sample Output

```
Iamneo@1 is a valid password
```

Sample Input

```
Iamneo
```

Sample Output

```
Iamneo is a invalid password
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q2. **Problem statement:**
Write a java program to check whether the given content is present in the pattern or not using the regex concept.

Input Format

The input consists of 2 strings. The first one is content and the second one is a pattern.

Output Format

The output prints the true or false. And with the content and pattern.

Sample Input

```
Iamneo test
.*test*.
```

Sample Output

```
Iamneo test contains .*test*. : true
```

Sample Input

```
I am a JAVA Programmer
.*the*.
```

Sample Output

```
I am a JAVA Programmer contains .*the*. : false
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q3. **Problem statement :**
Write a Java program to check whether a string contains only a certain set of characters (in this case a-z, A-Z, and 0-9)

Input Format

The input consists of strings.

Output Format

The output prints the input and returns the match whether it is True or False. Refer to the sample output for the formatting specifications.

Sample Input

```
ABCDEFabcdef123456
```

Sample Output

```
ABCDEFabcdef123456
true
```

Sample Input

```
W3.com
```

Sample Output

```
W3.com
false
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q4. **Problem statement:**
Write a Java program to check for a number at the end of a given string.

Input Format

The input consists of the string.

Output Format

The output prints the input text and results with whether the match is found or not found. Refer to the sample output for the formatting specifications.

Sample Input

```
abcdef
```

Sample Output

```
abcdef
Not matched!
```

Sample Input

```
abcdef3459
```

Sample Output

```
abcdef3459
Found a match!
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q5. **Problem statement:**
Write a Java program to count the number of vowels in a given string using regular expressions.

Input Format

The input consists of the string.

Output Format

The output consists of the original string and a new string with the count of the number of vowels. Refer to the sample output for the formatting specifications.

Sample Input

```
Java
```

Sample Output

```
Original string: Java
New string: 2
```

Sample Input

```
MID-CENTRALIZED
```

Sample Output

```
Original string: MID-CENTRALIZED
New string: 5
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q6. **Problem statement:**
Write a java program to find the number of occurrences of characters from the two strings.

Input Format

The input consists of two strings.

Output Format

The output prints the count of the number of occurrences. Refer to the sample input and output for the formatting specifications.

Sample Input

```
ab
abbbabbaba
```

Sample Output

```
The no of occurences: 3
```

Sample Input

```
a
aaabbbaaa
```

Sample Output

```
The no of occurences: 6
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q7. **Problem statement:**
Write a regular expression to represent all valid identifiers in java language.
Rules:
The allowed characters are:
1. a to z, A to Z, 0 to 9, -,#
2. The 1st character should be an alphabet symbol only.
3. The length of the identifier should be at least 2.

Sample Input

```
ashok
```

Sample Output

```
ashok:Valid Identifier
```

Sample Input

```
?ashok
```

Sample Output

```
?ashok:Invalid Identifier
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q8. **Problem statement:**
Write a regular expression to represent all mobile numbers.
1. Should contain exactly 10 digits.
2. The 1st digit should be 7 to 9.

Input Format

The input consists of digits.

Output Format

The output prints the mobile number by checking is it a valid number or an invalid number. Refer to the sample input and output for formatting specifications.

Sample Input

```
9989123456
```

Sample Output

```
9989123456 : Valid Number
```

Sample Input

```
6989654321
```

Sample Output

```
6989654321 : Invalid Number
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q9. **Problem statement:**
Write a java program finding data type of user input using Regular Expression.

Input Format

The input consists of different inputs of integer, string, double and date with different formats.
Date formats :

1. dd/mm/yyyy : eg: 01/12/2022
2. mm/dd/yyyy : eg: 12/24/2022
3. dd-mon-yy : eg: 01-apr-22
4. dd-mon-yyyy : eg: 01-apr-2022
5. dd-month-yy: eg:01-april-22
6. dd-month-yyyy: eg: 01-april-2022

Output Format

The output prints the value with the appropriate datatype. Refer to the sample input and output for the formatting specifications.

Sample Input

```
100
```

Sample Output

```
The datatype of 100 is: java.lang.Integer
```

Sample Input

```
52.87
```

Sample Output

```
The datatype of 52.87 is: java.lang.Double
```

Sample Input

```
21-apr-1994
```

Sample Output

```
The datatype of 21-apr-1994 is: java.util.Date
```

Sample Input

```
Born to win
```

Sample Output

```
The datatype of Born to win is: java.lang.String
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q10. **Problem statement:**
Write a Java Program to Extract a Single Quote Enclosed String From a Larger String using Regex.

Input Format

The input consists of two strings with single quotes.

Output Format

The output prints the string of the extracted string from the single quote. Refer to the sample input and output for the formatting specifications.

Sample Input

```
Finish what you 'start'
I will  'Finish'
```

Sample Output

```
First Extracted part: start
Second Extracted part: Finish
```

Sample Input

```
Action speaks  louder than 'words'
Action 'speaks'
```

Sample Output

```
First Extracted part: words
Second Extracted part: speaks
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Answer Key & Solution

Section 1 - CODING

Q1

Test Case

Input

```
Jav@21
```

Output

```
Jav@21 is a invalid password
```

Weightage - 25

Input

```
javar@g2
```

Output

```
javar@g2 is a invalid password
```

Weightage - 25

Input

```
Jav@re@1
```

Output

```
Jav@re@1 is a valid password
```

Weightage - 25

Input

```
J@v@re@3
```

Output

```
J@v@re@3 is a valid password
```

Weightage - 25

Sample Input

```
Iamneo@1
```

Sample Output

```
Iamneo@1 is a valid password
```

Sample Input

```
Iamneo
```

Sample Output

```
Iamneo is a invalid password
```

Solution

```
import java.util.Scanner;
class Main
{
    public Main()
    {
        super();
    }
    public static void main(String[] args)
```

```
{
       Main m = new Main();
        Scanner s = new Scanner(System.in);
        String passWord = s.nextLine();
        m.passwordValidation(passWord);

}
 public void passwordValidation(String password)
{
    boolean valid = true;
    if (password.length() > 15 || password.length() < 8)
    {

    valid = false;
    }
    String upperCaseChars = "(.*[A-Z].*)";
    if (!password.matches(upperCaseChars ))
    {
         valid = false;
    }
    String lowerCaseChars = "(.*[a-z].*)";
    if (!password.matches(lowerCaseChars ))
    {
        valid = false;
    }
    String numbers = "(.*[0-9].*)";
    if (!password.matches(numbers ))
    {
        valid = false;
    }
    String specialChars = "(.*[,~,!,@,#,$,%,^,&,*,(,),-,_,=,+,[,{,],},|,;,:,<,>,/,?].*$)";
    if (!password.matches(specialChars ))
    {
         valid = false;
    }
    if (valid)
    {
         System.out.println(password +" is a valid password");
    }
    else
            System.out.println(password + " is a invalid password");

    }
}
```

Q2

Test Case

Input

```
Iamneo test
.*Iamneo*.
```

Output

```
Iamneo test contains .*Iamneo*. : false
```

Weightage - 25

Input

```
Kind heart person
.*person*.
```

Output

```
Kind heart person contains .*person*. : true
```

Weightage - 25

Input

```
regular expression
.*regus*.
```

Output

```
regular expression    contains .*regus*. : false
```

Weightage - 25

Input

```
Intel Core
.*Core*.
```

Output

```
Intel Core contains .*Core*. : true
```

Weightage - 25

Sample Input

```
Iamneo test
.*test*.
```

Sample Output

```
Iamneo test contains .*test*. : true
```

Sample Input

```
I am a JAVA Programmer
.*the*.
```

Sample Output

```
I am a JAVA Programmer contains .*the*. : false
```

Solution

```
import java.util.*;
import java.util.regex.*;
class Main
{
   public static void main(String args[])
   {
        String content;
        Scanner s = new Scanner(System.in);
        content=s.nextLine();
        String pat;
          pat=s.nextLine();
      boolean isMatch = Pattern.matches(pat, content);
      System.out.println(content + " contains " +pat+" : "+ isMatch);
   }
}
```

Q3

Test Case

Input

```
SQL
```

Output

```
SQL
true
```

Weightage - 25

Input

Output

Java	Java true

Weightage - 25

Input	Output
www3.com	www3.com false

Weightage - 25

Input	Output
!@#$	!@#$ false

Weightage - 25

Sample Input	Sample Output
ABCDEFabcdef123456	ABCDEFabcdef123456 true

Sample Input	Sample Output
W3.com	W3.com false

Solution

```
import java.util.Scanner;
class Main
{
   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
       String str;
       str=s.nextLine();
       System.out.println(str);
       System.out.println(validate(str));
   }
   public static boolean validate(String text)
   {
    return text.matches("^[\\w]+$");
  }
}
```

Q4

Test Case

Input	Output
3abcdef	3abcdef Not matched!

Weightage - 25

Input

```
abcdef9
```

Output

```
abcdef9
Found a match!
```

Weightage - 25

Input

```
ghijklm
```

Output

```
ghijklm
Not matched!
```

Weightage - 25

Input

```
defg21494
```

Output

```
defg21494
Found a match!
```

Weightage - 25

Sample Input

```
abcdef
```

Sample Output

```
abcdef
Not matched!
```

Sample Input

```
abcdef3459
```

Sample Output

```
abcdef3459
Found a match!
```

Solution

```
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
class Main
{

   public static void main(String[] args)
   {

       Scanner  s =  new Scanner(System.in);
       String str;
       str=s.nextLine();
       System.out.println(str);
       System.out.println(validate(str));
   }

   public static String validate(String text)
   {
        Pattern pattern = Pattern.compile(".*[0-9]$");
```

```
        Matcher m = pattern.matcher(text);

        if (m.find())
            return "Found a match!";
        else
            return "Not matched!";
    }
}
```

Q5

Test Case

Input

```
LOWERED
```

Output

```
Original string: LOWERED
New string: 3
```

Weightage - 25

Input

```
Iamneo
```

Output

```
Original string: Iamneo
New string: 4
```

Weightage - 25

Input

```
Finishwhatyoustart
```

Output

```
Original string: Finishwhatyoustart
New string: 6
```

Weightage - 25

Input

```
MICROSOFT
```

Output

```
Original string: MICROSOFT
New string: 3
```

Weightage - 25

Sample Input

```
Java
```

Sample Output

```
Original string: Java
New string: 2
```

Sample Input

```
MID-CENTRALIZED
```

Sample Output

```
Original string: MID-CENTRALIZED
New string: 5
```

Solution

```
import java.util.*;
class Main
```

```
{
   public static void main(String[] args)
   {
        String text; //= "C++";
        Scanner  s = new Scanner(System.in);
        text=s.nextLine();
        System.out.println("Original string: "+text);
        System.out.println("New string: "+validate(text));
   }
   public static int validate(String text)
   {
       return text.replaceAll("[^aeiouAEIOU]", "").length();
   }
}
```

Q6

Test Case

Input

```
la
NeoColab
```

Output

```
The no of occurences: 1
```

Weightage - 25

Input

```
patn
pattern
```

Output

```
The no of occurences: 0
```

Weightage - 25

Input

```
core
intelcore
```

Output

```
The no of occurences: 1
```

Weightage - 25

Input

```
soft
Microsoft
```

Output

```
The no of occurences: 1
```

Weightage - 25

Sample Input

```
ab
abbbabbaba
```

Sample Output

```
The no of occurences: 3
```

Sample Input

```
a
aaabbbaaa
```

Sample Output

```
The no of occurences: 6
```

Solution

```
import java.util.regex.*;
import java.util.*;
class Main
{
    public static void main(String[] args)
    {
        int count=0;
        String str1,str2;
        Scanner  s =new Scanner(System.in);
        str1= s.nextLine();
        str2=s.nextLine();
        Pattern p=Pattern.compile(str1);//ab
        Matcher m=p.matcher(str2);//abbbabbaba
        while(m.find())
        {
            count++;
            //System.out.println(m.start()+"------"+m.end()+"------"+m.group());
        }
        System.out.println("The no of occurences: "+count);
     }
}
```

Q7

Test Case

Input

```
_number
```

Output

```
_number:Invalid  Identifier
```

Weightage - 25

Input

```
NUMBER
```

Output

```
NUMBER:Valid  Identifier
```

Weightage - 25

Input

```
NumBer
```

Output

```
NumBer:Valid  Identifier
```

Weightage - 25

Input

```
123abc
```

Output

```
123abc:Invalid  Identifier
```

Weightage - 25

Sample Input

```
ashok
```

Sample Output

```
ashok:Valid Identifier
```

Sample Input

```
?ashok
```

Sample Output

```
?ashok:Invalid Identifier
```

Solution

```
import java.util.regex.*;
import java.util.*;
class Main
{
    public static void main(String[] args)
    {
        Pattern p=Pattern.compile("[a-zA-Z][a-zA-Z0-9-#]+");
        Scanner  s =  new Scanner(System.in);
        String str;
        str=s.nextLine();

        Matcher m=p.matcher(str);

        if(m.find()&&m.group().equals(str))
        {
            System.out.print(str+":"+"Valid Identifier");
        }
        else
        {
         System.out.print(str+":"+"Invalid Identifier");
        }
    }
}
```

Q8

Test Case

Input

```
9989123456
```

Output

```
9989123456 : Valid Number
```

Weightage - 25

Input

```
09989123456
```

Output

```
09989123456 : Invalid Number
```

Weightage - 25

Input

```
919989123456
```

Output

```
919989123456 : Invalid Number
```

Weightage - 25

Input | **Output**

```
9989123456
```

```
9989123456 : Valid Number
```

Weightage - 25

Sample Input | **Sample Output**

```
9989123456
```

```
9989123456 : Valid Number
```

Sample Input | **Sample Output**

```
6989654321
```

```
6989654321 : Invalid Number
```

Solution

```
import java.util.regex.*;
import java.util.*;
class Main
{
    public static void main(String[] args)
    {
         Pattern p=Pattern.compile("[7-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]");
         Scanner  s = new  Scanner(System.in);
         String   str;
         str= s.nextLine();
         Matcher m=p.matcher(str);
         if(m.find()&&m.group().equals(str))
         {
            System.out.println(str+" : "+"Valid Number");
         }
         else
         {
            System.out.println(str+" : "+"Invalid Number");
         }
     }
}
//Pattern p=Pattern.compile("[7-9][0-9]{10}");
```

Q9

Test Case

Input | **Output**

```
21/04/1991
```

```
The datatype of 21/04/1991 is: java.util.Date
```

Weightage - 25

Input

```
04/21/1991
```

Output

```
The datatype of 04/21/1991 is: java.util.Date
```

Weightage - 25

Input

```
21-apr-92
```

Output

```
The datatype of 21-apr-92 is: java.util.Date
```

Weightage - 25

Input

```
21-apr-1994
```

Output

```
The datatype of 21-apr-1994 is: java.util.Date
```

Weightage - 15

Input

```
Core
```

Output

```
The datatype of Core is: java.lang.String
```

Weightage - 10

Sample Input

```
100
```

Sample Output

```
The datatype of 100 is: java.lang.Integer
```

Sample Input

```
52.87
```

Sample Output

```
The datatype of 52.87 is: java.lang.Double
```

Sample Input

```
21-apr-1994
```

Sample Output

```
The datatype of 21-apr-1994 is: java.util.Date
```

Sample Input

```
Born to win
```

Sample Output

```
The datatype of Born to win is: java.lang.String
```

Solution

```
import java.util.*;
class Main
{
    public static void main(String[] arg)
    {

        String input; //= "56.73";
        Scanner s =  new Scanner(System.in);
        input = s.nextLine();
        String dataType = null;

        // checking for Integer
        if (input.matches("\\d+"))
        {
            dataType = "java.lang.Integer";
        }

        // checking for floating point numbers
        else if (input.matches("\\d*[.]\\d+"))
        {
            dataType = "java.lang.Double";
        }

        // checking for date format dd/mm/yyyy
        else if (input.matches( "\\d{2}[/]\\d{2}[/]\\d{4}"))
        {
            dataType = "java.util.Date";
        }

        // checking for date format mm/dd/yyyy
        else if (input.matches("\\d{2}[/]\\d{2}[/]\\d{4}"))
        {
            dataType = "java.util.Date";
        }

        // checking for date format dd-mon-yy
        else if (input.matches("\\d{2}[-]\\w{3}[-]\\d{2}"))
        {
            dataType = "java.util.Date";
        }

        // checking for date format dd-mon-yyyy
        else if (input.matches( "\\d{2}[-]\\w{3}[-]\\d{4}"))
        {
            dataType = "java.util.Date";
        }

        // checking for date format dd-month-yy
        else if (input.matches("\\d{2}[-]\\w+[-]\\d{2}"))
        {
            dataType = "java.util.Date";
        }

        // checking for date format dd-month-yyyy
        else if (input.matches("\\d{2}[-]\\w+[-]\\d{4}"))
        {
            dataType = "java.util.Date";
        }

        // checking for date format yyyy-mm-dd
        else if (input.matches( "\\d{4}[-]\\d{2}[-]\\d{2}"))
        {
            dataType = "java.util.Date";
        }
```

```
        // checking for String
        else
        {
            dataType = "java.lang.String";
        }

        System.out.println("The datatype of " + input + " is: " + dataType);
    }
}
```

Q10

Test Case

Input

```
The early 'bird' catches the worm.
The 'early' bird catches the worm.
```

Output

```
First Extracted part: bird
Second Extracted part: early
```

Weightage - 25

Input

```
Don't judge  a book by its 'cover'
Don't  'judge'
```

Output

```
First Extracted part: cover
Second Extracted part: judge
```

Weightage - 25

Input

```
Better 'late'
than 'never'
```

Output

```
First Extracted part: late
Second Extracted part: never
```

Weightage - 25

Input

```
You 'catch' more flies
with 'honey' than with vinegar.
```

Output

```
First Extracted part: catch
Second Extracted part: honey
```

Weightage - 25

Sample Input

```
Finish what you 'start'
I will  'Finish'
```

Sample Output

```
First Extracted part: start
Second Extracted part: Finish
```

Sample Input

```
Action speaks  louder than 'words'
Action 'speaks'
```

Sample Output

```
First Extracted part: words
Second Extracted part: speaks
```

Solution

```
import  java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
class Main
{

    public static void main(String[] args)
    {
        String str1;
        Scanner  s = new Scanner(System.in);
        str1=s.nextLine();
        String str2;
        str2=s.nextLine();
        Pattern p = Pattern.compile(".*'([^']*)'.*");
        Matcher m1 = p.matcher(str1);
        Matcher m2 = p.matcher(str2);
        if (m1.matches())
        {
            System.out.println("First Extracted part: "+ m1.group(1));
        }
        if (m2.matches())
        {
            System.out.println("Second Extracted part: "+ m2.group(1));
        }
    }
}
```