# ASSIGNMENT 5

**Course Name: CS6308 Java Programming**
**Course Instructor: Jayachitra V P**
Date: 11/09/2024

## J51: Frequency Counter

**Write a program that first reads a piece of text entered by a user on one line, and then reads a key on the second line. The program displays the frequency with which the key has occurred in the piece of text.**

**EXAMPLE**

**INPUT**:

Can you write a whole paragraph without the letter a? Your sentences will sound wrong. Everyone will notice something weird. You will use uncommon words.

will

**OUTPUT:**

3

Explanation: The count of will is 3 in the text.

## J52: Palindrome

**Write a program that accepts a string from the user and prints whether it is a palindrome or not. Ignore the case of the characters.**

The format of the output is <input-string> <True/False>

**EXAMPLES:**

| | |
|---|---|
| **INPUT:** Nitin<br>**OUTPUT**: Nitin True | **INPUT:** Surya<br>**OUTPUT:** Surya False |

## J53. It’s Good

**Write a program as per the following specification: The input to the program is a string. The string contains substrings 'not' and 'bad' such that 'bad' comes after 'not'. There are only single occurrences of 'not' and 'bad'. The program outputs a string such that the whole 'not...bad' substring in the input is replaced by 'good'.**

| | |
|---|---|
| **INPUT:** Food is not bad.<br>**OUTPUT:** Food is good. | **INPUT:** The lyrics are not that bad!<br>**OUTPUT:** The lyrics are good! |

NOTE: In this question, all input strings for evaluation will definitely contain 'not' and 'bad' as substrings, such that 'bad' comes after 'not'.

**EXAMPLES:**

Write a program as per the following specification: The input to the program is a string.

The string may contain substrings 'not' and 'bad'. There are either 0 or 1 occurrences of 'not' and 'bad'. If 'bad' comes after 'not', then the program outputs a string such that the whole 'not...bad' substring in the input is replaced by 'good'. Otherwise, it prints the original string itself.

NOTE: In this question, the input strings for evaluation may or may not contain the substrings 'not' and 'bad' as substrings. Even if the input contains both, it is not guaranteed that 'bad' comes after 'not'.

**EXAMPLES:**

| | | |
|---|---|---|
| **INPUT:** The song is good.<br>**OUTPUT:** The song is good. | **INPUT:** Food is bad? not at all.<br>**OUTPUT:** Food is bad? not at all. | **INPUT:** The lyrics are not that bad!<br>**OUTPUT:** The lyrics are good! |

## J54. Character Count.

**Write a program to print the frequency of characters in a string in the given format.**

**EXAMPLES:**

**Input:** www.google.com

**Output:** w:3, .:2, g:2, o:3, l:1, e:1, c:1, m:1

**Input:** abbac

**Output:** a:2, b:2, c:1

(Please note that the order of characters in the output does not matter as long as the corresponding counts are correct).

## J55. Pangram

**Write a program to check whether an input string is a pangram or not. Pangrams are words or sentences containing every letter of the alphabet at least once. Ignore the case of the characters.**

- If the input string is a Pangram, the output should be: Yes, the string is a pangram.
- If the string is not a Pangram, it should report the missing letters, in lowercase, in ORDER. See the Examples below.

**EXAMPLES:**

**INPUT:** The brown fox jump over the lazy dog

**OUTPUT:** No, the string is NOT a pangram. Missing letter(s) is(are) c, i, k, q, s.

**INPUT:** The quick brown fox jumps over the lazy dog

**OUTPUT:** Yes, the string is a pangram.

Hint: boolean[] alphabet = new boolean[26];

## J56. Complex Number

**Write a program to create a ComplexNumber class with the following features:**

- Two private double fields to represent the real and imaginary parts.
- A constructor to initialize the complex number.
- Getter methods for the real and imaginary parts.
- Methods to add, subtract, multiply, and divide complex numbers.
- An override of the toString() method to represent the complex number in the form "a + bi" OR "a -bi" OR "-a -bi" OR "-a +bi".
- An override of the equals() method to compare two complex numbers.
- write a main method to demonstrate the usage of this class.

**Hint:**

```
public ComplexNumber add(ComplexNumber other) {

    double newReal = this.real + other.real;

    double newImaginary = this.imaginary + other.imaginary;

    return new ComplexNumber(newReal, newImaginary);
```

**}**

**@Override**

**public boolean equals(Object obj) { ..}**

**@Override**

**public String toString() { …}**

**Example:**

**Input:**

```
ComplexNumber c1 = new ComplexNumber(3, 4);
ComplexNumber c2 = new ComplexNumber(1, -2);
System.out.println("c1: " + c1);
System.out.println("c2: " + c2);
ComplexNumber sum = c1.add(c2);
ComplexNumber difference = c1.subtract(c2);
ComplexNumber product = c1.multiply(c2);
ComplexNumber quotient = c1.divide(c2);
boolean isEqual = c1.equals(c2);
System.out.println("c1 equals c2: " + isEqual);
```

**Output:**

```
c1: 3.0 + 4.0i
c2: 1.0 + -2.0i
Sum: 4.0 + 2.0i
Difference: 2.0 + 6.0i
Product: 11.0 + -2.0i
Quotient: -0.5 + 1.5i
c1 equals c2: false
```