**Data Structure and Algorithm**

**Lab 1: ArrayList and Linked List**

**Candidate: Imraul Kayes Emmaka**

**Student ID No: 201739060027**

**Major: Computer Science**

**Supervisor: Shan He**

**School of Computer Science**

**Southwest Petroleum University**

**§1 Exercises**

1. The next four questions refer to the following String elements:

["It", "was", "a", "stormy", "night"]

(1) Write the code to declare an ArrayList containing these elements.

(2) Write code to insert two additional elements, "dark" and "and", at the proper places in the list to produce the following ArrayList as the result: ["It", "was", "a", "dark", "and", "stormy", "night"]

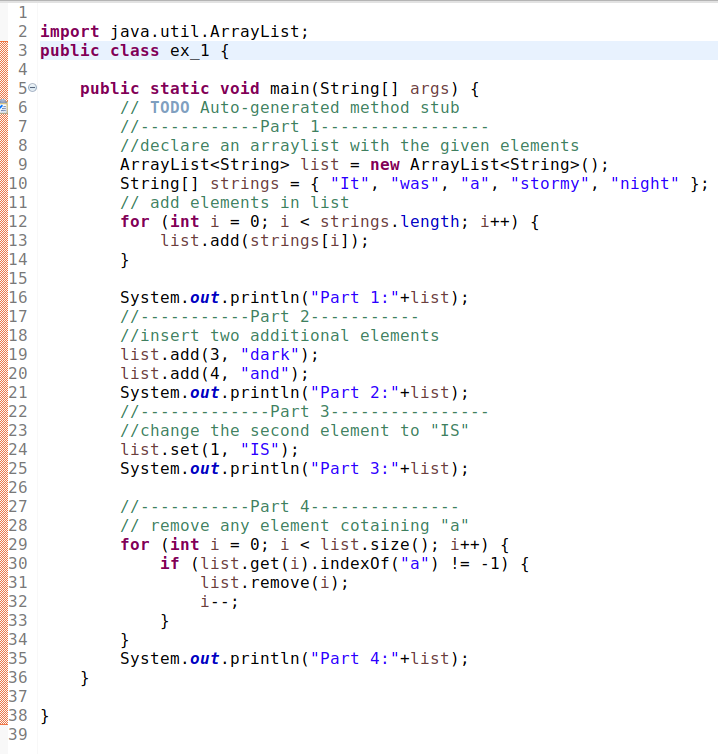
(3) Write code to change the second element’s value to "IS", producing the following ArrayList as the result:

["It", "IS", "a", "dark", "and", "stormy", "night"]

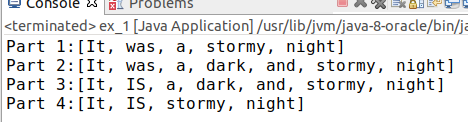
(4) Write code to remove from the list any Strings that contain the letter "a". The following should be the list’s contents after your code has executed:

["It", "IS", "stormy", "night"]

**Source code:**

****

**Result:**

****

2. Suppose you have an input file of names that has some duplicates, and you want to get rid of the duplicates. The file might look like this:

Maria Derek Erica

Livia Jack Anita

Kendall Maria Livia Derek

Jamie Jack

Erica

**Source code:**

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Scanner;

**public** **class** ex\_2\_2 {

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

Scanner sc = **new** Scanner(**new** File("names.txt"));

// declare an ArrayList names to store the names from the file

ArrayList<String> words = **new** ArrayList<String>();

ArrayList<String> noDuplicateStrings = **new** ArrayList<String>();

// add names to the ArrayList names

**while** (sc.hasNext()) {

String name = sc.useDelimiter("[^a-zA-Z']+").next().toLowerCase();

words.add(name);

}

// sort names

Collections.*sort*(words);

String name = words.get(0);

**for** (**int** i = 1; i < words.size(); i++) {

**if** (!words.get(i).equals(name)) {

noDuplicateStrings.add(name);

}

name = words.get(i);

**if** (i == words.size() - 1) {

noDuplicateStrings.add(name);

}

}

System.***out***.println(noDuplicateStrings);

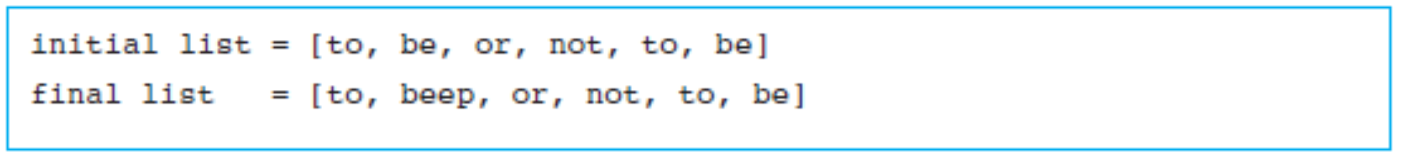
}

}

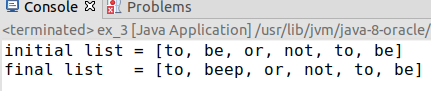
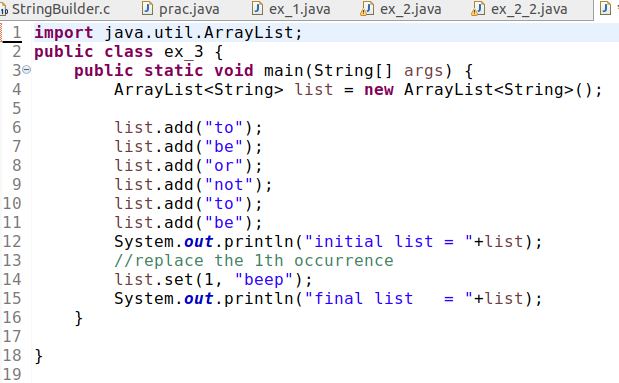
**Result:**

****

3. Write code to produces the following output (Replace the first occurrence of one word in an ArrayList<String> with another word):



**Source code:**

**Result:**

4. Search engines like Google ignore the stop words in users’ queries. The idea is that certain words like “a” and “the” appear so often that they aren’t worth indexing.

Google’s full list of stop words is believed to have at least 35 entries, but we’ll settle for 15 of the most obvious choices. To explore how removing stop words can affect a text, our program will read a file called speech.txt that contains the first part of Hamlet’s famous speech:

To be or not to be – that is the question:

Whether 'tis nobler in the mind to suffer

The slings and arrows of outrageous fortune

Or to take arms against a sea of troubles,

And by opposing end them.

**Source code:**

import java.util.\*;

import java.io.\*;

public class problem\_4 {

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

Scanner sc = new Scanner(new File("speech.txt"));

//declare an ArrayList stopWords to store the stop words

ArrayList stopWords = new ArrayList();

//add stop words into stopWords

stopWords.add("a");

stopWords.add("be");

stopWords.add("by");

stopWords.add("how");

stopWords.add("in");

stopWords.add("is");

stopWords.add("it");

stopWords.add("of");

stopWords.add("on");

stopWords.add("or");

stopWords.add("that");

stopWords.add("the");

stopWords.add("this");

stopWords.add("to");

stopWords.add("why");

System.out.println(stopWords);

System.out.println(stopWords.size());

//declare an ArrayList speech to store the speech

ArrayList speech = new ArrayList();

//add elements in speech

while (sc.hasNext()){

speech.add(sc.next().toLowerCase());

}

//remove the stopWords.

//Method - 1:

// speech.removeAll(stopWords);

// System.out.println(speech);

// System.out.println(speech.size());

//Method - 2:

for (int i = 0; i < stopWords.size(); i++){

for(int j = 0; j < speech.size(); j++){

if(stopWords.get(i).equals(speech.get(j))){

speech.remove(j);

j = j - 1;

}

}

}

//Method - 3:

// for(int i = 0; i < stopWords.size(); i++){

// if (speech.contains(stopWords.get(i))){

// speech.remove(speech.indexOf(stopWords.get(i)));

// i = i-1;

// }

// }

//print speech after removing the stopWords and its size.

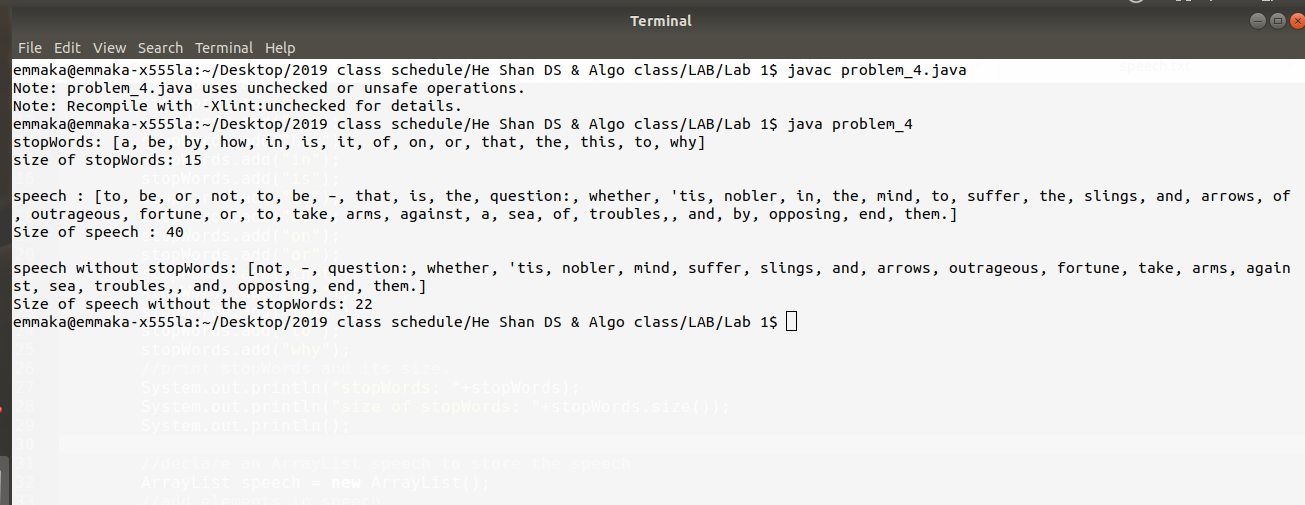
System.out.println("speech without stopWords: "+speech);

System.out.println("Size of speech without the stopWords: "speech.size());

}

}

**Result:**

****

5. Write a piece of code that counts the number of duplicate elements in a linked list, that is, the number of elements whose values are repeated at an earlier index in the list. Assume that all duplicates in the list occur consecutively. For example, the list (1, 1, 3, 5, 5, 5, 5, 7, 7, 11) contains five duplicates: one duplicate of element value 1, three duplicates of element value 5, and one duplicate of element value 7.

**Source code:**

**import** java.util.LinkedList;

**public** **class** problem\_5 {

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

LinkedList<Integer> list = **new** LinkedList<Integer>();

list.add(1);

list.add(1);

list.add(3);

list.add(5);

list.add(5);

list.add(5);

list.add(5);

list.add(7);

list.add(7);

list.add(11);

// ArrayList count = new ArrayList<>();

// ArrayList key = new ArrayList();

//find the max of list

**int** max = list.get(list.size()-1);

System.***out***.println(max);

//count the number of duplicates

**int**[] count = **new** **int**[max];

**int** prev = (**int**) list.get(0);

**for** (**int** i = 1; i < list.size(); i++) {

**int** pres = (**int**) list.get(i);

**if** (prev == pres) {

count[prev]++;

}

prev = pres;

}

//print the number of duplicates of the elements in list

**for** (**int** i = 1; i < count.length; i++) {

**if** (count[i] != 0)

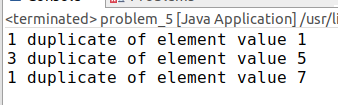
System.***out***.println(count[i] +" duplicate of element value "+i);

}

}

**}**

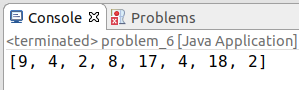
**result:**

****

6. (Optional) Write a method called removeAll that accepts a linked list of integers as a parameter and removes all occurrences of a particular value. You must preserve the original relative order of the remaining elements of the list. For example, the call removeAll(list, 3) would change the list (3, 9, 4, 2, 3, 8, 17, 4, 3, 18, 2, 3) to (9, 4,2, 8, 17, 4, 18, 2).

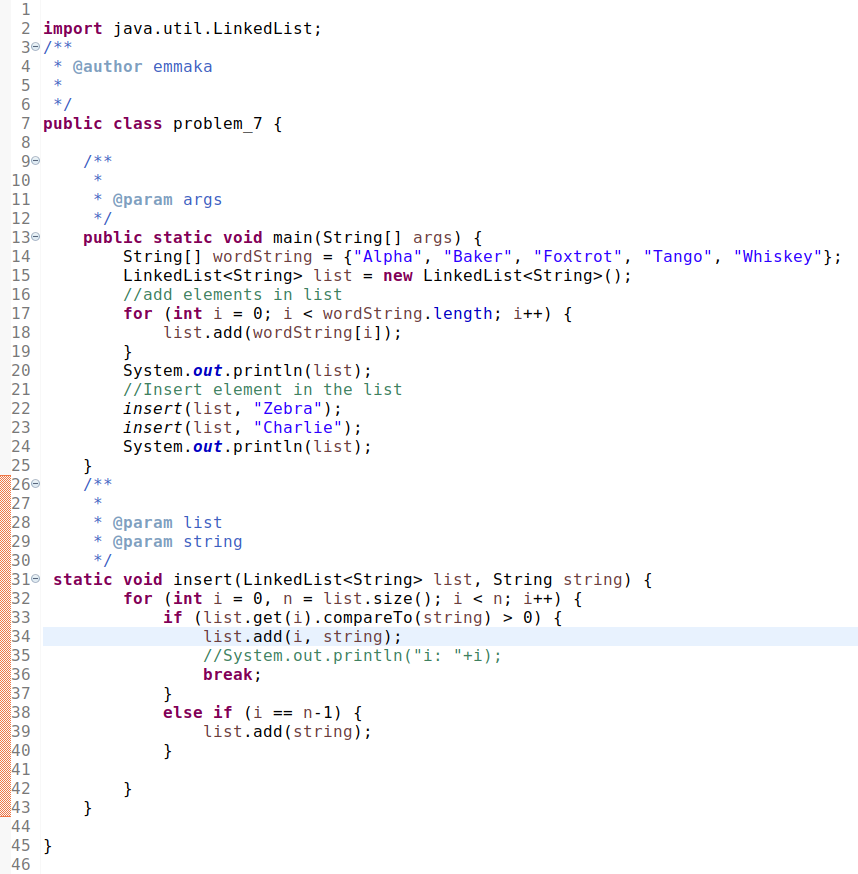
**Source code:**



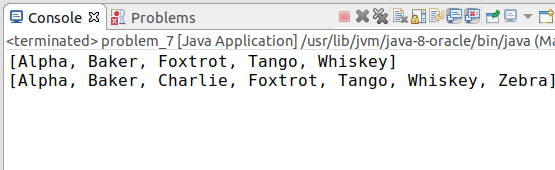
**Result:**

7. (Optional) Write a piece of code that inserts a String into an ordered linked list of Strings, maintaining sorted order. For example, for the list ("Alpha", "Baker", "Foxtrot", "Tango", "Whiskey"), inserting "Charlie" in order would produce the list ("Alpha", "Baker", "Charlie", "Foxtrot", "Tango", "Whiskey").

**Source code:**

****

**Result:**

****

**§2Runtime Environment**

**Software platform: Windows (Vista or Higher) ,** **jGRASP Version 2.0.4 or above,** **jdk-10.0.1 or above**

**§3 Program List and Experimental Result**

**§4 Summary**