**Problem 1)** Average Value (Java 8 Lambdas and Streams) .Write a method that returns the average of a list of integers.

public Double average(List<Integer> list) {

return list.stream().mapToInt(i -> i).average().getAsDouble();

}

**Problem 2)** Comma Separated (Java 8 Lambdas and Streams)

Write a method that returns a comma separated string based on a given list of integers. Each element should preceded by the letter 'e' if the number is even, and preceded by the letter 'o' if the number is odd. For example, if the input list is (3,44), the output should be 'o3,e44'.

public String getString(List<Integer> list) {

return list.stream().map(i -> i % 2 == 0 ? "e" + i : "o" + i).collect(joining(","));

}

**Problem 3)** Filter Strings (Java 8 Lambdas and Streams)

Given a list of Strings, write a method that returns a list of all strings that start with the letter 'a' (lower case) and have exactly 3 letters. TIP: Use Java 8 Lambdas and Streams API's.

public List<String> search(List<String> list) {

return list.stream().filter(s -> s.startsWith("a")).filter(s -> s.length() == 3).collect(Collectors.toList());

}

**Problem 4)**

1. Basic lambdas. Make an array containing a few Strings. Sort it by

• length (i.e., shortest to longest) **-**

• reverse length (i.e., longest to shortest) **-**

• alphabetically by the first character only **-**

(Hint: charAt(0) returns the numeric code for the first character)

• Strings that contain “e” first, everything else second. For now, put the code directly in the lambda.

(Hint: remember that the body of a lambda are allowed to have curly braces and a return statement.

See the first two lambda examples in the notes.) **-**

• Redo the previous problem, but use a static helper method so that your lambda looks like this:

Arrays.sort(words, (s1,s2) -> Utils.yourMethod(s1,s2)) **-**

String[] words = {"e", "end", "guts", "heat", "coding"};

Arrays.sort(words, (s,r) -> s.length() - r.length());

System.out.println("Ascending order of length: " + Arrays.toString(words));

Arrays.sort(words, (s,r) -> r.length() - s.length());

System.out.println("Descending order of length: " + Arrays.toString(words));

Arrays.sort(words);

System.out.println("Alphabetical order: " + Arrays.toString(words));

Arrays.sort(words, Comparator.comparingInt(i -> (i.contains("e") || i.contains("E") ? 1 : 0)));

System.out.print("Strings that contain “e” first: "+ Arrays.toString(words));

**Notes on the sorting problems Problem 4:**

• The compare method of Comparator should return a negative number if the first entry is “less”

than the second, a positive number if the first entry is “greater” than the second, and 0 if they are

the same. For details, see http://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html.

• To print out an array after sorting, do System.out.println(Arrays.asList(yourArray))

The point of this is that if you just print an array directly, you do not see anything useful (just the

memory address), but if you print a List, it shows the individual elements separated by commas.

So, the above trick is simpler than making a loop to traverse the array and print out the elements.

Note for future reference that this trick only works if yourArray is an Object array (e.g., String[]

or Integer[]); this trick fails if yourArray is an array of primitives (e.g., int[]).