EECS2030 Fall 2018 **Name: (Last, First)** _____
Advanced OOP
Lab Test 2 Exercise Solution
Time Limit: 80 minutes **Student ID** _____

---

1 Programming Exercises

- Download and import this starter project.
- This exercise is about a simple application of 2-D points, each with the x- and y-coordinates. A line contains its start and end points. A line collector stores a collection of points. Points in a point collector are *positioned* according to their chronological order of being added (i.e., first-added point at position 0, second-added point at position 1, and so on). Two points are equal if their coordinate values are qual. Two lines are equal if their start and end points are equal. Two line collectors are equal if they store the same number of points, and points at the corresponding positions are equal.
- As before, you are **only** allowed to use primitive arrays.
- There are two packages (`aggreagation` and `composition`), each of which containing a JUnit test class for you to study and infer the necessary classes and methods.
- Before you start this programming exercise, make sure you understand the difference between aggregation (where sharing and thus aliasing is allowed) and composition (where sharing is forbidden).
- You are advised **not to** dive into coding right away and just focus on passing the assertions. The actual test will be similar but with a different example, so merely passing all tests in the exercise (without understanding how things work together) will not lead you far.
- Instead, first study the tests assertions in the two packages: they are very similar, but the differences indicate the distinct requirements of aggregation and composition (in particular the expected results from using == versus `equals` to compare objects).

2 Written Exercises

1. Consider the following statements:

   (**A**) $3n + 7$ is $O(n \cdot log(n))$
   (**B**) $3n + 7$ is $O(n)$
   (**C**) $3n + 7$ is $O(1)$
   (**D**) $3n + 7$ is $O(2^n)$
   (**E**) $3n + 7$ is $O(log(n))$
   (**F**) $3n + 7$ is $O(n^2)$

   (a) Which of the above statement or statements are *correct*?

   Solution: Statements **A** **B** **D** **F**

   [of 10 marks]

   (b) Among the above statement or statements that are *correct*, which one is the most *accurate*?

   Solution: Statement **B**

   [of 5 marks]

   (c) Justify your answer to the previous question. That is, clearly explain why it is more *accurate* than all other *correct* statements.

   Solution: The highest power of n in $3n + 7$ is one. So Statement B is the most accurate by saying that $3n + 7$ is $O(n)$. The class $O(n)$ is strictly contained by $O(n \cdot log(n))$, which is strictly contained by $O(n^2)$, which is strictly contained by $O(2^n)$.

   [of 10 marks]

2. In order to prove that $f(n) = 4n^3 - 5n^2 + 59 + n^4 + 9n$ is $O(n^4)$, you need to choose values for two constants: constant c as a factor for n^4 and constant n_0 as some starting value of n.

   (a) Write down the precise condition for which c and n_0 must satisfy in order for the proof to succeed. **Hint:** Your answer should involve n^4, $f(n)$, c, and n_0.

   Solution:
   $$c \cdot n^4 \geq f(n) \quad \text{for } n \geq n_0$$

   [of 5 marks]

   (b) Give values of c and n_0 that will complete the proof.

   Solution: Choose $c = 78$ and $n_0 = 1$.

   [of 5 marks]

3. Consider the following Java program:

```
1  duplicatePrint(int[] a, int n)
2    int sum;
3    for (int i = 0; i < n; i++) {
4      for (int j = 0; j < i; j++) {
5        for (int k = 0; k < 5; k++) {
6          System.out.println(a[k]);
7        }
8      }
9    }
```

Determine the **most accurate** asymptotic upper bound of the above program, using the big-Oh notation. You **must** show in detail how you determine the bound. Without a convincing derivation process, you will only receive partial marks.

Solution:

- From Line 3, we know that the body of the outer loop will run n times.
- From Line 4, we know that:
  - 1st iteration of the outer loop, body of the middle loop will run 0 time.
  - 2nd iteration of the outer loop, body of the middle loop will run 1 times.
  - 3rd iteration of the outer loop, body of the middle loop will run 2 times.
  - ...
  - $(n-1)_{th}$ iteration of the outer loop, body of the middle loop will run $(n-2)$ times.
  - n_{th} iteration of the outer loop, body of the middle loop will run $(n-1)$ times.

  middle loop
- That is, the body of the middle loop will run in total:

$$0+1+2+\cdots+(n-2)+(n-1)=\frac{n(n-1)}{2}$$

  which is $O(n^2)$
- From Line 5, we know that body of the inner most loop will be executed for exactly 5 times.
- Line 6 is a primitive operation that requires some constant running time, say c.
- Therefore, the running time of the above algorithm is $O(n^2 \cdot 5 \cdot c)$ which is $O(n^2)$.

[of 15 marks]