Homework 2 – Due Sep. 24th 23:59, KST (GMT+9)

Instructions: Complete the report and turn it in before the due date. Any deviations from the instructed deliverable format will result in a deduction of grade. DO NOT COPY OTHER'S WORKS!

The answers to the following questions must be accompanied by appropriate justification. You will receive no credit if you do not show your work (drawing a graph does not count).

1. Compute the tightest order, in big-O notation, of the following functions and codes. In all cases, the variable to consider is 'n'. (4 pts x 5 = 20 pts)

   A. $\mathrm{f(n)} = 4\mathrm{n}^3 - 100\mathrm{n}^2 + 0.01\mathrm{n}^4$

   B. $\mathrm{f(n)} = \frac{100000000\sqrt{n}}{n} + 2n^{-2} + \frac{1}{n}$

   C. $\mathrm{f(n)} = \cos(n)+4$

   D. $\mathrm{f(n)} = \sin^2(n)$

   E. **for**(**int** i = 0; i * i < n; i++) System.out.println(i);

2. Attach your code for homework 1's HW1.len() and HW1.addStudent(Student). Based on YOUR code, analyze the big-O complexity of the given methods (If you didn't do HW1, write it now). The style of analysis should closely follow what we did in class for the insertion sort. (10 pts x 2 = 20 pts)

3. Fill in the body of the following Java method. This method should return the minimum value of the given sorted array in constant time. Note that the array can be sorted in either ascending or descending order. After writing the code, provide an analysis similar to what you did in problem #2 to show your implementation does indeed run in O(1).
   (20 pts)

```
public static double getMinimum(int[] arr) {
    // Fill in here (arr is sorted in either ascending or descending order)
}
```

4. Fill in the body of the following Java method. This method scans the given array 'arr' with 'item' and returns the number of elements in 'arr' that are less than or equal to 'item'.
   (e.g., If arr = {1, 2, 3, 6, 0, -1, 5} and item = 5, then count(arr, item) should return 6)
   Provide a complexity analysis on count() as you did in #3, except now do it for two cases: best case and worst case (20 pts x 2 = 40 pts). For an extra credit of 5 pts, perform the analysis on the average case as well. You should clearly identify what and why an average case is what you claim it is.

```
public static int count(int[] arr, int item) {
        // Return the # of elements that are <= item
}
```

Deliverables

- A single HW2-<your_name>.pdf file containing your name and SBU ID. The PDF file must contain all the answers to the problems above in a ***typed*** format. That is, don't hand-write your answers and attach the scanned version – it's very hard to discern poorly handwritten text. Do not turn in .java files for the coding problems: just include the code in your final PDF file.