Lab 1

1. Decide whether each of the following is true or false.

   a. $4n^3 + n$ is $\Theta(n^3)$.

   b. $\log n$ is $o(n)$.

   c. 2^n is $\omega(n^2)$.

2. Determine the asymptotic running time of the following procedure (an exact number of primitive operations is not necessary):

```
int[] arrays(int n) {
   int[] arr = new int[n];
   for(int i = 0; i < n; ++i){
      arr[i] = 1;

   }
   for(int i = 0; i < n; ++i) {
      for(int j = i; j < n; ++j){
         arr[i] += arr[j] + i + j;
      }
   }
   return arr;

}
```

3. Consider the following problem: As input you are given two sorted arrays of integers. Your objective is to design an algorithm that would merge the two arrays together to form a new sorted array that contains all the integers contained in the two arrays. For example, on input

```
[1, 4, 5, 8, 17], [2, 4, 8, 11, 13, 21, 23, 25]
```

   the algorithm would output the following array:

```
[1,2,4,4,5,8,8, 11, 13, 17, 21, 23, 25]
```

   For this problem, do the following:

   A. Design an algorithm `Merge` to solve this problem and write your algorithm description using the pseudo-code syntax discussed in class.

   B. Examining your pseudo-code, determine the asymptotic running time of this merge algorithm

   C. Implement your pseudo-code as a Java method `merge` having the following signature:

```
int[] merge(int[] arr1, int[] arr2)
```

   Be sure to test your method in a `main` method to be sure it really works!

4. Below, pseudo-code is given for the recursive factorial algorithm `recursiveFactorial`. Determine the asymptotic running time of this algorithm.

```
Algorithm recursiveFactorial(n)
   Input: A non-negative integer n
   Output: n!
    if (n = 0 || n = 1) then
         return 1
    return n * recursiveFactorial(n-1)
```

5. *Interview Question.* You are given a length-n array A consisting of 0s and 1s, arranged in sorted order. Give an Little-o(n) algorithm that counts the total number of 0s and 1s in the array. Your algorithm may not make use of auxiliary storage such as arrays or hashtables (more precisely, the only additional space used, beyond the given array, is O(1)). Explain why your algorithm runs in Little-o(n) time.