Lab 2

1. Determine the asymptotic running time of the following procedure (an exact computation of number of basic 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;

}
```

2. 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!

3. **Big-oh and Little-oh.** Use the definitions of $O(f(n))$ and limit facts about $o(f(n))$ given in class to decide whether each of the following is true or false, and in each case, prove your answer.

   A. $1 + 4n^2$ is $O(n^2)$
   B. $n^2 - 2n$ is *not* $O(n)$
   C. $\log(n)$ is $o(n)$
   D. n is *not* $o(n)$

4. **Power Set Algorithm**. Given a set X, the power set of X, denoted P(X), is the set of all subsets of X. Below, you are given an algorithm for computing the power set of a given set. This algorithm is used in the brute-force solution to the SubsetSum Problem, discussed in the first lecture. Implement this algorithm in a Java method:

   ```
   List powerSet(List X)
   ```

   Use the following pseudo-code to guide development of your code

   Algorithm: PowerSet(X)

   Input: A list X of elements
   Output: A list P consisting of all subsets of X – elements of P are *Sets*

   ```
   P ← new list
   S ← new Set     //S is the empty set
   P.add(S)        //P is now the set { S }
   T  ← new Set
   while (!X.isEmpty() ) do
       f  ← X.removeFirst()
       for each x in P do
           T ← x ∪ {f}   // T is the set containing f & all elements of x
           P.add(T)
   return P
   ```