This homework is due at the start of class on Wednesday, February 5.

1. Suppose that A and B are integer arrays of size n. Give a run time analysis for each of the following Java code segments. That is, find a function $f(n)$ such that the run time is in $\Theta(f(n))$. Briefly justify your answers.

a)
```
for (int i = 0; i < n; i++) {
    B[i] = 0;
    for (int j = 0; j < i; j++)
        B[i] = B[i] + A[j];
}
```

b)
```
int[] C = new int[n];
for (int i = 0; i < n; i++) {
    C[i] = A[i]*B[i];
}
```

2. Suppose that A is an integer array of length n. Consider the following algorithm:

```
for (int i = 0; i < n-1; i++) {
    for(int j = i+1; j < n; j++) {
        if (A[i] == A[j]) {
            return true;
        }
    }
}
return false;
```

a) What problem does this algorithm solve? (That is, what is the meaning of the return value?)

b) What is the *worst-case* run time analysis for the algorithm? What sort of array gives the worst-case run time? What is the *best-case* run time analyis? What sort of array gives the best-case run time?

c) Devise a new algorithm that solves the same problem. The improved algorithm should start by *sorting* the array. What is the run time analysis of the improved algorithm?

3. Devise an algorithm that finds the second-largest element in an array A of length N. You can assume that $N \geq 2$ and that all the elements in the array are different. (Write the algorithm either in Java or in careful pseudocode.) What is the run-time analysis for your algorithm? Exactly how many comparisons of array elements does it do? (That is, say what you can about the actual number of comparisons, not just a Big-Oh analysis.)

4. B is an infinite array of numbers, **sorted into increasing order**. You know that the array contains the number X, but you don't know where in the array it is located. Devise an algorithm that will find the index, N, of X in the array (that is, find N such that $A[N] = X$). **Your algorithm must have run time $\Theta(log(N))$, where N is the index of X in the array.**

5. Apply the Master Theorem to solve the following recurrance relations:

a)
```
T(1) = 1
T(n) = 9*T(n/3) + n^2
```

b)
```
T(1) = 1
T(n) = 9*T(n/3) + n^3
```

c)
```
T(1) = 1
T(n) = 5*T(n/4) + n
```