Due **Thursday, July 11th** at the beginning of class.

1. Determine the basic operation(s) in the following algorithm and then analyze its performance

```
for i = 1 to n do
{
     for j = i to n do
          x++;
     for j = 1 to i do
          y++;
}
```

2. Update the following algorithm so that it reduced from $\Theta(n)$ to $\Theta(1)$

```
sum = 0;
for i = 1 to n do
     sum = sum + i;
```

3. Verify whether each of the following pairs of functions are in the same complexity class. If they are not, indicate which one grows faster.

a. $\lg n$ and $\lg\sqrt{n}$
b. $(\lg n)^2$ and $\lg n^2$
c. $100n^2$ and $0.01n^3$

4. Suppose an algorithm with complexity $\Theta(n^3)$ takes 10 minutes for a problem with input size 100. What size problem would we expect to be able to solve in 270 minutes?

5. Rank the following functions by order of growth from low to high

(n - 2)! 2^{2n} $0.001n^4 + 3n^3 + 1$ 3^n $\sqrt[3]{n}$

6. Write an algorithm that finds the *m* smallest numbers in a list of *n* numbers

7. Show directly that $f(n) = n^2 + 3n^3 \in \Theta(n^3)$. We do this by showing that it is both big O of n^3 and Omega of n^3.

8. Consider the following algorithm:

```
int add_them (int n, int A[])
{
     index i, j, k;
     j = 0;
     for (i = 1; i <= n; i++)
          j = j + A[i];
     k = 1;
     for (i = 1; i <= n; i++)
          k = k + k;
     return j + k;
}
```

What is the time complexity T(n) of the algorithm? Try to improve the efficiency of the algorithm.

9. Consider the following algorithm:

```
int any_equal (int n, int A[][])
{
     index i, j, k, m;
     for ( i = 1; i <= n; i++)
          for ( j = 1; j <= n; j++)
               for ( k = 1; k <= n; k ++)
                    for ( m = 1; m <= n; m++)
                         if (A[i][j] == A[k][m] && !( i==k && j ==m))
                              return 1;
     return 0;
}
```

What is the best case time complexity of the algorithm (assuming n > 1)?
What is the worst case time complexity of the algorithm?