1) For each of the following, show that $f \in O(g)$. That is, you will need to find values for $c$ and $n_0$ such that the definition of big-O holds true as we did with the examples in lecture.

a) $f(n) = 12n$ \quad $g(n) = \frac{n}{5}$

b) $f(n) = 6n^2 + 1000$ \quad $g(n) = n^4$

c) $f(n) = 6\log(n)$ \quad $g(n) = .5n$
2) For each of the following program fragments, determine the asymptotic runtime in terms of n

a)
public void mysteryOne(int n) {
    int x = 0;
    for (int i = n; i >= 0; i--)
    {
        if ((i % 5) == 0) {
            break;
        } else {
            for (int j = 1; j < n; j *= 2) {
                x++;
            }
        }
    }
}

b)
public void mysteryTwo(int n) {
    int x = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = i; j < ((n * n - 1)/3); j++)
        {
            x += j;
        }
    }
}

c)
public void mysteryThree(int n) {
    for (int i = 0; i < n; i++)
    {
        methodTwo(i);
    }
}

private void methodTwo(int x) {
    if (x > 0) {
        methodTwo(x - 1);
    }
}
3) For each of the following, determine if \( f \in \Theta(g), f \in \Omega(g), f \in O(g) \), several of these, or none of these.

a) \( f(n) = \log n \quad g(n) = \log \log n \)

b) \( f(n) = 2^n \quad g(n) = 2^n \)

c) \( f(n) = 25n^3 \quad g(n) = n^3 + 25n \)
4) Pseudocode and recurrence relations
a) Write pseudocode for a function that calculates the largest difference between any two numbers in an array of positive integers with a runtime in $\Theta(n^3)$.

For example, the largest difference between any two numbers in the following array would be 19.

\[ a = [4, 6, 3, 9, 2, 1, 20] \]

b) Can this function be written with a runtime in $\Theta(n)$? If yes, write pseudocode below. If no, why? What would have to be different about the input in order to do so?

c) Can this function be written with a runtime in $\Theta(1)$? If yes, write pseudocode below. If no, why? What would have to be different about the input in order to do so?
5) Recurrence Relations

a) Find the tightest Big-Oh bound for the following recurrence relation $T(n) = n + T(n/2)$. Justify your answer.

b) Find a Big-Oh bound for the following recurrence relation $T(n) = n + 2T(n/2)$. Justify your answer.
6) Growth Rates
a) Order the following functions from slowest to fastest growth rate

• $2^7$
• $n^2 \log n$
• $2^{n/2}$
• $\log n$
• $n \log n^2$
• $n^6$
• $n \log \log n$
• $n \log^2 n$
• $n$
• $n^2$
• $n \log n$
• $2^n$
• $\log^2 n$
• $2/n$
• $n^{1/2}$
7) Big-Oh Definition
Suppose $T_1(n)$ is $O(f(n))$ and $T_2(n)$ is $O(f(n))$. Which of the following are always true (for all $T_1$, $f$, and $T_2$)?
You do not need to prove an item is true (just saying true is enough for full credit), but if an item is false, you need to give a counterexample to demonstrate it is false. To give a counterexample, give values for $T_1(n)$, $T_2(n)$, and $f(n)$ for which the statement is false (for example, you could write, “The statement is false if $T_1(n) = 100n$, $T_2(n) = 2n^2$, and $f(n) = n^3$”). Hints: Think about the definitions of big-$O$, big-$\Omega$, and big-$\Theta$.

a) $T_1(n)/T_2(n)$ is $O(1)$.

b) $T_1(n) + T_2(n)$ is $\Omega(f(n))$.

c) $T_1(n) - T_2(n)$ is $O(f(n))$.

d) $T_1(n)$ is $O(T_2(n))$.

e) $T_2(n)$ is $\Theta(T_1(n))$. 