MCS-360 (HW-10 sol^n/Rubric)

Problem 1.

```
i = 1;
t = 0;
while( i <= t ) {
    t = t + i;
    i = 2 * i;
}
```

The while loop is never executed since $1 > 0$.

So this segment runs in constant time, or $O(1)$.

(2pts)

Problem 2.

x^{2^k} is found by squaring x successively: $x \to x^2 \to (x^2)^2 \to \cdots$

Clearly, to find x^{2^k} starting at x, we need to square x, k times!

$$x \xrightarrow[1^{st}\text{ squaring}]{} x^{2^1} \xrightarrow[2^{nd}\text{ squaring}]{} x^{2^2} \xrightarrow[3^{rd}\text{ squaring}]{} x^{2^3} \to \cdots \xrightarrow[k^{th}\text{ squaring}]{} x^{2^k}.$$

So, the runtime of this algorithm would be $O(k)$, as it takes k steps to run.

(2pts) for $O(k)$

Recursive algorithm:

Let $f(x,k) = x^{2^k}$. Then we note that

$$\left(x^{2^{k-1}}\right)^2 = x^{2^{k-1}} \cdot x^{2^{k-1}} = x^{2^{k-1}+2^{k-1}} = x^{2 \cdot 2^{k-1}} = x^{2^k}.$$

Thus, $f(x,k) = f(x,k-1) * f(x,k-1)$. ← recursive relation.

And $f(x,0) = x^{2^0} = x^1 = x$ ← base case.

So, a C++ function to compute x^{2^k} would be:

```
1: double f( double x, int k ) {
2:     if (k==0) return x;
3:     else {
4:         double retVal = f(x,k-1);
5:         return retVal * retVal;
6:     }
7: }.
```

(2pts) for C++ code

Notes: (1) Instead of doing $f(x,k) = f(x,k-1)^2$ we could also use $f(x,k) = f(x^2,k-1)$.

(2) returning $f(x,k-1) * f(x,k-1)$ in Line 5 would cause the program to compute $f(x,k-1)$ twice, leading to an inefficient algorithm.

Problem 3.

For each of these problems, we only need to consider the dominant terms on the LHS.

(a) $3n^2 + 10n\log n \sim 3n^2 \neq O(n\log n)$. — False

(b) $3n^2 + 10n\log n \sim 3n^2 = \Omega(n^2)$ — True

(c) $3n^2 + 10n\log n \sim 3n^2 = \Theta(n^2)$ — True

(d) $n\log n + n/2 \sim n\log n \neq O(n)$ — False

(e) $10\sqrt{n} + \log n \sim 10\sqrt{n} = O(n)$ — True

(f) $\sqrt{n} + \log n \sim \sqrt{n} \neq O(\log n)$ — False

(g) $\sqrt{n} + \log n \sim \sqrt{n} \neq \Theta(\log n)$ — False

(h) $\sqrt{n} + \log n \sim \sqrt{n} \neq \Theta(n)$ — False

(i) $2\sqrt{n} + \log n \sim 2\sqrt{n} = \Theta(\sqrt{n})$ — True

(j) $\sqrt{n} + \log n \sim 2\sqrt{n} = \Omega(1)$ — True

(k) $\sqrt{n} + \log n \sim \sqrt{n} = \Omega(\log n)$ — True

(l) $\sqrt{n} + \log n \sim \sqrt{n} = \Omega(n)$ — False

(4 pts)
(0.33 pts for each part)

Note: When finding O, Ω, Θ, o, you may also ignore constant factors in front of the functions, for example $3n^2 + 10\log n \sim n^2$ for the purposes of comparing two functions and finding out whether their are O, Ω, Θ or o of one another.