1. Problem 0.1 on page 8 of the textbook. Write either $O$, $\Omega$ or $\Theta$ in each blank. Do not write $O$ or $\Omega$ if $\Theta$ is correct.

(a) $n - 100 = \Theta (n - 200)$

(b) $n^{1/2} = O (n^{2/3})$

(c) $100n + \log n = \Theta (n + \log^2 n)$

(d) $n \log n = \Omega (10n + \log(10n))$

(e) $\log(2n) = \Theta (\log(3n))$

(f) $10 \log n = \Theta (\log(n^2))$

(g) $n^{1.01} = \Omega (n \log^2 n)$

(h) $n^2/\log n = \Omega (n \log^2 n)$

(i) $n^{0.1} = \Omega (\log^2 n)$

(j) $(\log n)^{\log n} = \Omega (n/ \log n)$

(k) $\sqrt{n} = \Omega (\log^3 n)$

(l) $n^{1/2} = O (5^{\log_2 n})$

(m) $n2^n = O (3^n)$

(n) $2^n = \Theta (2^{n+1})$

(o) $n! = \Omega (2^n)$

(p) $\log_2 n^{\log_2 n} = O (2^{(\log_2 n)^2})$

(q) $\sum_{i=1}^{n} i^k = \Theta (n^{k+1})$

2. The following problem is a modified version of problem 0.3(c) on page 9 of the textbook. The answer is not exactly the same as the answer to the textbook version, but the techniques are similar.

$F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$. We start by assuming $F_n = n^C$ for some constant $C$. This is false, but it’s true in the limit; that is, for larger and larger values of $n$, the ratio of $F_n$ to
$C^n$ converges to 1. That is, there is a constant $C$ such that $\lim_{n \to \infty} \frac{F_n}{n^C} = 1$. for the correct value of $C$. Making the assumption that they are actually equal:

\[
\begin{align*}
F_{n+2} &= F_{n+1} + F_n \\
C^{n+2} &= C^{n+1} + C^n
\end{align*}
\]

Divide both sides by $C^n$:

\[
C^2 = C^1 + C^0
\]

Which is a quadratic equation over $C$. The quadratic formula gives us two solutions: $C = \frac{1 \pm \sqrt{5}}{2}$ But $C$ is not the negative solution, since $\{F_n\}$ would converge to zero. thus $C = \frac{1 + \sqrt{5}}{2}$ which is the so-called golden ratio. Since we did make an incorrect assumption, we might worry about this answer. But it’s correct.

3. Consider the following C++ program.

```cpp
void process(int n)
{
    cout << n << endl;
    if(n > 1) process(n/2);
    cout << n%2;
}

int main()
{
    int n;
    cout << "Enter a positive integer: ";
    cin >> n;
    assert(n > 0);
    process(n);
    cout << endl;
    return 1;
}
```

The last line of the output of `process(n)` is a string of bits. What does this bitstring represent?

The binary numeral for $n$. 
4. The C++ code below implements a function, “mystery.” What does it compute?

```cpp
float square(float x)
{
    return x*x;
}

float mystery(float x, int k)
{
    if (k == 0) return 1.0;
    else if (x == 0.0) return 0.0;
    else if (k < 0) return 1/mystery(x,-k);
    else if (k%2) return x*mystery(x,k-1);
    else return mystery(square(x),k/2);
}
```

It computes $x^k$.

5. Write the asymptotic time complexity for each code fragment, using Θ notation.

(a) for (int i=1; i < n; i++)
    for (int j=i; j > 0; j--)
    Θ(n^2)

(b) for (int i=1; i < n; i=2*i)
    for (int j=i; j < n; j++)
    Θ(n log n)

(c) for (int i=1; i < n; i = 2*i)
    for (int j=1; j < i; j++)
    Θ(n)

(d) for (int i=1; i < n; i++)
    for (int j=1; j < i; j = j*2)
    Θ(n log n)

(e) for (int i=1; i < n; i++)
    for (int j=i; j < n; j = j*2)
    Θ(n)

(f) for (int i=2; i < n; i = i*i)
    Θ(log log n)

(g) for (int i=1; i*i < n; i++)
    Θ(√n)

(h) for (int i=n; i > 1; i = i/2)
    for (int j=1; j < i; j=2*j)
    Θ(log^2 n)