1. Problem 0.1 on page 8 of the textbook. In each of the following situations, write $O$, $\Omega$, $\Theta$ in the blank.

(a)  $n - 100 = \Theta(n - 200)$
(b)  $n^{1/2} = O(n^{2/3})$
(c)  $100n + \log n = \Theta(n + \log^2 n)$
(d)  $n \log n = \Theta(10n \log(10n))$
     $n \log n = \Omega(10n + \log(10n))$
(e)  $\log(2n) = \Theta(\log(3m))$
(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)  $n^{2^n} = O(3^n)$
(n)  $2^n = \Theta(2^{n+1})$
(o)  $n! = \Omega(2^n)$
(p)  $\log n^{\log n} = O(2^{(\log_2 n)^2})$ [hard]
(q)  $\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$

2. Work problem 0.3(c) on page 9 of the textbook.

$F_n = F_{n-1} + F_{n-2}$ We start by assuming $F_n = 2^{nC}$ for some $C$. This is false, but its almost true, that is

$$\lim_{n \to \infty} \frac{F_n}{2^{nC}} = 1$$

for the correct value of $C$. Making that assumption:

$$F_{n+2} = F_{n+1} + F_n$$
$$2^{C(n+2)} = 2^{C(n+1)} + 2^{Cn}$$

Divide both sides by $2^{Cn}$:  $2^{2C} = 2^C + 2^0$

Substitute $x = 2^C$ :  $x^2 = x + 1$

By the quadratic formula, since $2^C > 0$ :  $x = \frac{1 + \sqrt{5}}{2}$ the golden ratio!

$$C = \log_2 \left( \frac{1 + \sqrt{5}}{2} \right)$$
3. For any positive integer input, say \( n \), the second column is a string of bits. What does that bitstring represent?

The binary numeral for \( n \), written in reverse.

4. Each of these code fragments takes \( O(n \log n) \) time, but not necessarily \( \Theta(n \log n) \). Give the asymptotic complexity of each in terms of \( n \), using \( \Theta \) in each case.

(a) 
```cpp
for(int i = 1; i < n; i++)
    for(int j = 1; j < i; j = 2*j);
    cout << "Hello" << endl;
\Theta(n \log n)
```

(b) 
```cpp
for(int i = 1; i < n; i++)
    for(int j = i; j < n; j = 2*j);
    cout << "Hello" << endl;
\Theta(n)
```

(c) 
```cpp
for(int i = 1; i < n; i=2*i)
    for(int j = 1; j < i; j++);
    cout << "Hello" << endl;
\Theta(n)
```

(d) 
```cpp
for(int i = 1; i < n; i=2*i)
    for(int j = i; j < n; j++);
    cout << "Hello" << endl;
\Theta(n \log n)
```

(e) 
```cpp
for(int i = n; i > 1; i=i/2)
    for(int j = i; j > 1; j--);
    cout << "Hello" << endl;
\Theta(n)
```

(f) 
```cpp
for(int i = n; i > 1; i=i/2)
    for(int j = n; j > i; j--);
    cout << "Hello" << endl;
\Theta(n \log n)
```

5. These problems are harder than the ones above. Given the asymptotic complexity of each fragment in terms of \( n \), using \( \Theta \).

(g) 
```cpp
for(int i = 1; i < n; i=2*i)
    for(int j = 1; j < i; j=2*j);
    cout << "Hello" << endl;
```
Hint: Use substitution. Let \( m = \log n \), \( k = \log i \), \( l = \log j \).

```cpp
for(int k = 0; k < m; k++)
    for(int l = 0; l < k; l++)
        cout << "Hello" << endl;
\Theta(m^2) = \Theta(\log^2 n)
```
(h) for(int i = 2; i < n; i=i*i) 
    cout << "Hello" << endl;
Hint: Use substitution. Let m = log n, k = log i.
for(int k = 1; k < m; k=2*k) 
    cout << "Hello" << endl;
Θ(log m) = Θ(log log n)

(i) for(int i = 2; i < n; i=i*i) 
    for(int j = 1; j < i; j = 2*j) 
        cout << "Hello" << endl;
Hint: Use substitution. Let m = log n, k = log i, l = log j.
for(int k = 1; k < m; k=2*k) 
    for(int l = 0; l < k; l++)
        Θ(m) = Θ(log n)

(j) for(int i = n; i > 1; i = log i) 
    cout << "Hello" << endl;
Hint: The answer is a function you’ve possibly never heard of. That function is defined on page 136 of the textbook. I will simply tell you the answer, and you will need to remember it.
Θ(log* n)