1.1

We know that f(n) = Θ(g(n)) and g(n) = Θ(h(n))

$\Theta(g(n)) = \{f(n): \exists c_1 > 0 \wedge c_2 > 0 \wedge n_0, 0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n), \forall n \geq n_0\}$ as A and

$\Theta(h(n)) = \{g(n): \exists c_1 > 0 \wedge c_2 > 0 \wedge n_0, 0 \leq c_1 * h(n) \leq g(n) \leq c_2 * h(n), \forall n \geq n_0\}$ as B

Simply, c1*g(n) ≤ f(n) ≤ c2*g(n) and c3*h(n) ≤ g(n) ≤ c4*h(n) when n>n0. We can combine the A and B, and we can get:

0 ≤ c1*c3*h(n) ≤ f(n) ≤ c2*c4*h(n) when n exceeds some point.

Let c5 = c1*c3 and c6 = c2*c4, and c5 and c6 both are constant

According to definition, f(n) = Θ(h(n)).

1.2

If f(n) = Θ(f($\frac{n}{2}$)), then we have $\Theta(f(\frac{n}{2})) = \{f(n): \exists c_1 > 0 \wedge c_2 > 0 \wedge n_0, 0 \leq c_1 * f(\frac{n}{2}) \leq f(n) \leq c_2 * f(\frac{n}{2}), \forall n \geq n_0\}$

Simply, $c1 * f\left(\frac{n}{2}\right) \leq f(n) \leq c2 * f\left(\frac{n}{2}\right)$ when n>n_0.

And we know that the $n = O(2^n)$, if f(n) = 2^n, and let g(n) = $f\left(\frac{n}{2}\right) = 2^{\frac{n}{2}}$.

$\frac{\lim_{n\to\infty} g(n)}{\lim_{n\to\infty} f(n)} = \lim_{n\to\infty} \frac{2^{\frac{n}{2}}}{2^n} = \lim_{n\to\infty} \frac{1}{2^{\frac{n}{2}}} = 0$. To conclude, f(n) is not $\Theta(f(\frac{n}{2}))$.

1.3

(a) $f(n) = kn * log(n)\ and\ g(n) = n * log(kn)$ for any positive constant k.

If k≥1:

$f(n) = kn * \log(n) = n\log\left(n^k\right), g(n) = n * \log(n * k)$

When c=1, let $f(n) \geq g(n), k * log(n) \geq \log(n) + \log(k)$

$$(k-1) * \log(n) \geq \log(k)$$

$$\log(n) \geq \frac{\log(k)}{k-1}$$

If the x is basic of log, $x^{\log_x n} \geq x^{\frac{\log(k)}{k-1}}$ and we can get: $n \geq k^{\frac{1}{k-1}}$.

So, f(n) = $\Omega(g(n))$. In other hand, $k * g(n) = kn * \log(n * k)$.

Because $k \geq 1$, $n * k \geq n$. So $k * g(n) \geq kn * \log(n) = f(n)$.

So, $k \geq 1, c = k, n_0 = 0, f(n) = O(g(n))$.

If k<1:

$k * g(n) = kn * \log(n * k)$.

Because $k < 1$, $n * k < n$. So $k * g(n) < kn * \log(n) = f(n)$.
So, $k < 1, c = k, n_0 = 0, f(n) = \Omega(g(n))$.

In other hand, when c=1, let $f(n) \leq g(n), k * log(n) \leq \log(n) + \log(k)$

$$(1-k) * \log(n) \geq \log(k)$$

$$\log(n) \geq \frac{\log(k)}{1-k}$$

If the x is basic of log, $x^{\log_x n} \geq x^{\frac{\log(k)}{1-k}}$ and we can get: $n \geq k^{\frac{1}{k-1}}$.

So, f(n) = $O(g(n))$
Above all, whether k≥0 or k<1, f(n) = Θ(g(n))

(b) $\mathrm{f(n) = n(\sin(n))^2}$ and $\mathrm{g(n)} = \sqrt{\mathrm{n}}$

In the f(n), the $\sin^2(n)$ $is \in [0,1], and\, n * \sin^2(n) \in [0, n]$.
However, the function is changed from time to time, we cannot know that what is the result when it limits to ∞. g(n) = $\sqrt{n}$, let h(n)=f(n)-g(n)$= \mathrm{n(\sin(n))^2} - \mathrm{c}\sqrt{\mathrm{n}}$, the range of h(n) is $[-\mathrm{c}\sqrt{\mathrm{n}}$, n], we cannot find a constant c to always make h(n) >= 0 or <=0. So there are not relationship between f(n) and g(n).

(c) $f(n) = \sqrt{n}^{\sqrt{n}} and\ g(n) = \sqrt{n^n}$

$f(n) = n^{\frac{\sqrt{n}}{2}}\ and\ g(n) = n^{\frac{n}{2}}$

And log two sizes: $logf(n) = \frac{\sqrt{n}}{2}\log(n), \log g(n) = (\frac{n}{2})log(n)$

They have the same in log(n), so we just compare the n and $\sqrt{n}$
When $n \geq 1$, $n \geq \sqrt{n}$. To conclude, c = 1 and $n_0 = 1$, f(n) = O(g(n)).

(d) $f(n) = n^{\log(\log(n))}\ and\ g(n) = log(n)^{log(n)}$

$$\log(f(n)) = \log(\log(n)) * \log(n),\ \log(g(n)) = \log(\log(n)) * \log(n)$$

We can find that log(f(n)) equal to log(g(n)).
Therefore, f(n) = Θ(g(n)) and g(n) = Θ(f(n)).

(e) $f(n) = \Sigma_{i=1}^{n} i^2\ and\ g(n) = n * \Sigma_{i=1}^{n}(n-i)$

f(n)=$\frac{1}{6}n(n+1)(2n+1) = \frac{1}{6}(2n^3 + 3n^2 + n)$;

g(n)=$n * \frac{(n-1)n}{2} = \frac{n^3 - n^2}{2}$ (Sum arithmetic sequence) if $c = 1$, we assume that $\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n \le \frac{1}{2}n^3 - \frac{1}{2}n^2$;

Solve this function $2n^3 + 3n^2 + n \le 3n^3 - 3n^2$

$$n^3 - 6n^2 - n \ge 0$$
$$n^2 - 6n - 1 \ge 0$$

We can calculate the result $n_0 \approx 6.162$, because we know that n is an integer, which means that n ≥ 7, f(n)≥g(n), c=1, so f(n) = $\Omega(g(n))$

If c = $\frac{3}{2}$, the function will lose the term to the third power.

Let the g(n) sizes equal to each other, we can get and solve:

$\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n \ge \frac{2}{3}(\frac{1}{2}n^3 - \frac{1}{2}n^2)$

$$\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n \ge \frac{1}{3}n^3 - \frac{1}{3}n^2$$
$$\frac{1}{2}n^3 + \frac{3}{4}n^2 + \frac{1}{4}n \ge \frac{1}{2}n^3 - \frac{1}{2}n^2$$
$$\frac{5}{4}n^2 + \frac{1}{4}n \ge 0$$
$$\frac{5}{4}n + \frac{1}{4} \ge 0$$

So, when $n \ge 0$, c*g(n) ≥ f(n), c = $\frac{2}{3}$. f(n) = $O(g(n))$

Above all, f(n) = Θ(g(n))

2.1

T(n) = $k * T\left(\frac{n}{2}\right) + \log_2\left(\frac{n}{2}\right)$ when $n > 1$ and T(n) = 1

The Case 1, k ≠1:

T(n) $= k * T\left(\frac{n}{2}\right) + \log_2\left(\frac{n}{2}\right)$

$$= k * \left(kT\left(\frac{n}{4}\right) + \log_2\left(\frac{n}{4}\right)\right) + \log_2\left(\frac{n}{2}\right)$$
$$= k * (k * \left(kT\left(\frac{8}{n}\right) + \log_2 \frac{8}{n}\right) + \log_2\left(\frac{n}{4}\right)) + \log_2 (\frac{n}{2})$$
$$= \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$= k^i * T\left(\frac{n}{2^i}\right) + k^{i-1} * \log_2\left(\frac{n}{2^i}\right) + \ \ldots\ldots\ldots + k^0 * \log_2 \frac{n}{2^1}$$

The i is the times of the traversal, and the k is just a value.
Then we can solve the following equation:

$$k^{i-1} * \log_2\left(\frac{n}{2^i}\right) + \ \ldots\ldots\ldots + k^0 * \log_2 \frac{n}{2^1}$$
$$= k^{i-1}(\log_2(n) - i) + k^{i-2}(\log_2(n) - i - 1) + \ldots + k^0(\log_2(n) - 1)$$

$= (k^{i-1} + k^{i-2} + ...k^1 + k^0) * \log_2 n - (k^{i-1} * i + ... + k^1 * 2 + 1)$

$= \log_2 n * \frac{(k^i - 1)}{k-1} - \frac{(i * k^{i+1} - (i+1) * k^{i+1})}{(k-1)^2}$

let i = $\log_2 n$,

T(n) = $\log_2 \left(\frac{k^{\log_2 n} - 1}{k-1}\right) - \frac{(\log_2 n * k^{\log_2 n + 1} - (\log_2 n + 1) * k^{\log_2 n} + 1)}{(k-1)^2} + k^{\log_2 n} * T\left(\frac{n}{2^{\log_2 n}}\right)$

The tight asymptotic for T(n) is $\Theta(logn * k^{\log_2 n})$ when k$\neq 1$

The Case 2, k=1:

T(n) $= k * T\left(\frac{n}{2}\right)$ + $\log_2\left(\frac{n}{2}\right)$

$= k * \left(kT\left(\frac{n}{4}\right) + \log_2\left(\frac{n}{4}\right)\right) + \log_2\left(\frac{n}{2}\right)$

$= k * (k * \left(kT\left(\frac{8}{n}\right) + \log_2 \frac{8}{n}\right) + \log_2\left(\frac{n}{4}\right)) + \log_2 (\frac{n}{2})$

$= \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$(k=1)

$= T\left(\frac{n}{2^i}\right) + \log_2\left(\frac{n}{2^i}\right) +$ .... $+ \log_2 \frac{n}{2^1}$

Then we can solve the following equation:

$\log_2\left(\frac{n}{2^i}\right) +$ .... $+ \log_2 \frac{n}{2^1}$

$= (\log_2 (n) - i) + (\log_2 (n) - i - 1) + ... + (\log_2 (n) - 1)$

$= i * \log_2 n + (i + (i-1)... + 2 + 1)$

$= i * \log_2 n + \frac{(i+1)*i}{2}$

let i = $\log_2 n$,

T(n) = $\log_2 n * \log_2 n + \frac{(\log_2 n + 1) * \log_2 n}{2} + T\left(\frac{n}{2^{\log_2 n}}\right)$

The tight asymptotic for T(n) is $\Theta((\log_2 n)^2)$ when k$= 1$

2.2

Through induction, we know that the time of the recursion is $\log_2 n + 1$. For the base case, the if statement should be executed every time and complexity is 1, and the return statement will be executed only one time. In other case, if $\frac{n}{2}$ is an even, then it will take four complexity (if statement is 2 and the return statement is 2).

Else it will take five complexity (the if statement is 2 and return statement is 3).

```
int pow(int n) {
    if(n == 0)       ---------------------------------------   1
        return 1;      ----------------------------------   1
```

```
    int pmid = pow(n / 2);        ---------------------      1, recursion times: log2 n + 1
    if(n % 2 == 0) {        -----------------------------     2
        return pmid * pmid;        -------------------     2
    } else {
        return 2 * pmid * pmid;        ---------------     3
    }
}
```

At first, if statement takes one time and the time of the recursion by the if is $\log_2 n + 1$, return 1 is only one time. All steps are the even number case is smallest case (like n = 8), and 1 is odd, the later if statement is 1+2+3 = 6. The even time is $\log_2 n - 1$, the complexity is 4+1 (assign pmid at a time). To conclude, the sum of them is $\log_2 n + 1 + 1 + 6 + (\log_2 n - 1) * 5 + 5 = 6 * \log_2 n$ + 8

All steps are the odd number case is smallest case (like n = 7). the later if statement is 1+2+3 = 6. There is not even time, the complexity is 4+1 (assign pmid at a time). To conclude, the sum of them is $\log_2 n + 1 + 1 + 1 + 6 + (\log_2 n) * 6 = 7 * \log_2 n$ +9.

To conclude, the result is $T\left(\frac{n}{2}\right)$+ 8 ≤ T(n) ≤ $T\left(\frac{n}{2}\right)$ + 9.

2.3

We know that T(n) = Θ(log(n)). In the function, we can find that the n is not changed every times. It will divide to the left for a while and divide to the right for a while, and the loop times is $\log_2 n$.

We should also pay attention that the recursion is in the condition of the loop. So we can get that:

U(n)=$\log_2 n$*(T(n)+O(1)).

$U(n) = \log_2 n$*$(T(n) + O(1))$

$U\left(\frac{n}{2}\right) = \log_2 \frac{n}{2} * (T\left(\frac{n}{2}\right) + O(1))$

$$U(n) - U\left(\frac{n}{2}\right)$$

$$= \log_2 n * T(n) + \log_2 n * O(1) - \log_2 \frac{n}{2} * T\left(\frac{n}{2}\right) - \log_2 \frac{n}{2} * O(1))$$

$$= \log_2 2 + \log_2 n * \theta(\log n) - \log_2 \frac{n}{2} * \theta(\log n)$$

$$= \log_2 2 + \log_2 2 * \theta(\log n)$$

So, $U(n) = U\left(\frac{n}{2}\right) + \theta(\log n)$, c = $\log_2 2$.