1. (a) Consider $c := 15 + 20 + 2015 = 2050$ and $n_0 := 1$. Then $0 \le 15n^3 + 20n^2 \log n + 2015 \le 15n^3 + 20n^3 + 2015n^3 \le cn^3$ for all $n \ge n_0$. (Note that $\log n < n$ for $n \ge 1$.)

   (b) Consider $c := 1$ and $n_0 := 1024$. Then, using the fact that $(\log n)^3 < n$ for all $n \ge 1024$, we have $0 \le cn(\log n)^3 < n^2$ for $n \ge n_0$.

   (c) We have:

   - the upper bound: $8n - n^2/(n-200) \le 8n$ for $n \ge 200$.
   - the lower bound: Note that $n^2/(n-200) = 2n(n/(2n-400)) \le 2n$ for $n \ge 200$. So, $8n - n^2/(n-200) \ge 8n - 2n = 6n$.

   Thus, for $c_1 = 6$, $c_2 = 8$ and $n_0 = 200$ we have $0 \le c_1 n \le 8n - n^2/(n-200) \le c_2 n$ for $n \ge n_0$.

   (d) Let $c > 0$ be given. We should find n_0 so that for $n \ge n_0$ we have $2^n > cn^{50}$, i.e., $n > \log c + 50 \log n$ which implies $n - 50 \log n > \log c$. Assuming $n \ge 1024$, we have $\log n < n/100$ and subsequently $n - 50 \log n > n/2$. So, in order to have $n - 50 \log n > \log c$, it suffices to have $n/2 > \log c$, i.e., for $n > n_0 = \max\{1024, 2 \log c\}$, we have $2^n > cn^{50}$.

   (e) Let $c > 0$ be given. We should find n_0 so that for $n \ge n_0$, we have $1395n < cn \log n$, i.e., $1395 < c \log n$ which implies $n > 2^{1395/c}$. So, it suffices to have $n_0 = \max\{1, 2^{1395/c}\}$.

2. (a) $f(n) \in o(g(n))$. Either show directly that the definition of o is is satisfied, or show that $\lim_{n\to\infty} f(n)/g(n) = 0$.

   (b) $f(n) \in \omega(g(n))$. Either show directly that the definition of Ω is satisfied, or show that $\lim_{n\to\infty} f(n)/g(n) = \infty$.

   (c) $f(n) \in \omega(g(n))$. Probably taking derivatives is the easiest. Following definition or comparing growth rates by replacing $\log \log n$ with m (and taking derivatives) are also acceptable.

   (d) $f(n) \in \Theta(g(n))$. It suffices to show that both $f(n)$ and $g(n)$ are $\Theta(n^3)$. For that, we note that $4 - 2(\cos n)^3$ is in the range $[2, 6]$.

3. (a) False. Counter example: Consider $f(n) := n$ and $g(n) := \begin{cases} 1 & n \text{ odd} \\ n^2 & n \text{ even} \end{cases}$. To prove the claim false it will be sufficient to show that $f(n) \notin O(g(n))$ and $f(n) \notin \Omega(g(n))$, since then the antecedent of the implication is satisfied while the consequent is not.

   If $f(n) \in O(g(n))$, then there exist constants $n_0 > 0$ and $c > 0$ such that $f(n) \le cg(n)$ for all $n \ge n_0$. But for any odd number $n_1 > c$ we have $f(n_1) = n_1 > c = cg(n_1)$, showing that $f(n) \notin O(g(n))$.

   Similarly, if $f(n) \in \Omega(g(n))$, then there exists constants $n_0 > 0$ and $c > 0$ such that $cg(n) \le f(n)$ for all $n \ge n_0$. But for any even number $n_1 > 1/c$ we have $cg(n_1) = cn_1^2 > n_1 = f(n_1)$, showing that $f(n) \notin \Omega(g(n))$.

(b) True. Proof: Assume that $f(n) \in \Theta(g(n))$ and $h(n) \in \Theta(g(n))$, and let $n_1, n_2 > 0$ and $c_1, c_2, c_3, c_4 > 0$ be such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_1$ and $c_3 g(n) \leq h(n) \leq c_4 g(n)$ for all $n \geq n_2$. Since g and h are positive, for every $n \geq n_2$ we have

$$\frac{1}{c_4 g(n)} \leq \frac{1}{h(n)} \leq \frac{1}{c_3 g(n)}.$$

Let $n_0 = \max\{n_1, n_2\}$. Then for every $n \geq n_0$ we have

$$\frac{c_1 g(n)}{c_4 g(n)} \leq \frac{f(n)}{h(n)} \leq \frac{c_2 g(n)}{c_3 g(n)} \Rightarrow \frac{c_1}{c_4} \leq \frac{f(n)}{h(n)} \leq \frac{c_2}{c_3}.$$

Selecting constants $c_1' = \frac{c_1}{c_4}$ and $c_2' = \frac{c_2}{c_3}$ we have $c_1' \leq \frac{f(n)}{h(n)} \leq c_2'$ for every $n \geq n_0$. Thus, according to the definition of Θ we have $\frac{f(n)}{h(n)} \in \Theta(1)$.

(c) True. We will show that $f(n)g(n)/(f(n)+g(n)) \leq \min(f(n), g(n)) \leq 2f(n)g(n)/(f(n)+g(n))$ for all $n \geq 1$. The desired result will then follow from the definition of Θ using $c_1 = 1$, $c_2 = 2$ and $n_0 = 1$.

For brevity, let f denote $f(n)$ and g denote $g(n)$, $n \geq n_0$. By assumption, f and g are positive, so $fg/(f+g) = \min(f,g)\max(f,g)/(f+g)$, which is less than $\min(f,g)$ since $\max(f,g)/(f+g) < 1$. Similarly, $\min(f,g) = 2fg/(2\max(f,g)) \leq 2fg/(f+g)$.

(d) False. Counter example: Consider $f(n) = \log_3 n$ and $g(n) = 2\log_3 n$. Then $f(n) \in \Theta(g(n))$ but $3^{f(n)} = n$ and $3^{g(n)} = n^2$.

4. The WHILE loop is repeated $\log(n)$ times. So we get the following recursion:

$$F(n) = c_1 + c_2 \log(n) + F(1) + F(2) + \ldots + F(n/2) \quad (n > 1)$$

Rewriting this for $F(n/2)$:

$$F(n/2) = c_1 + c_2 \log(n) - c_2 + F(1) + F(2) + \ldots + F(n/4) \quad (n > 1)$$

The different of the above two equalities is:

$$F(n) - F(n/2) = F(n/2) + c_2 \quad (n > 1)$$

The resulting recursion is $F(n) = 2F(n/2) + c_2$ which can be solved to get $\Theta(n)$ (a simple justification for that is required).

5. (a) $\sum_{i=1}^{n^3} \sum_{j=1}^{i^3} c = \sum_{i=1}^{n^3} (ci^3) = n^3 c + (n^3 \times (n^3+1)/2)^2 \in \Theta(n^{12})$

(b) Before the second loop, the value of p is 2^i. The time complexity is $\sum_{i=1}^{n} \sum_{j=1}^{2^i} c = \sum_{i=1}^{n} c \cdot 2^i = c \cdot (2^{n+1} - 2) \in \Theta(2^n)$.

(c) $\sum_{i=1}^{3n} (c_1 + \sum_{j=1395}^{2015} \sum_{k=4i}^{6i} c_2) = \sum_{i=1}^{3n} (c_1 + \sum_{j=1395}^{2015} 2ic_2)$
$= \sum_{i=1}^{3n} (c_1 + 1240ic_2) = 3nc_1 + 1240 \cdot 3n(3n+1)/2 \in \Theta(n^2)$

(d) In each two iterations of the while loop, the value of s is at least divided by 2, and at most divided by 16. The time complexity is at most $\log_2(3n) \in O(\log n)$ and at least $\log_{16}(3n) \in \Omega(\log n)$. So, it is $\Theta((n)$.

6. (a) If the root is larger than c, do nothing. Otherwise, report the root and recurs on its left and right children. The indices that are checked are the report elements and (potentially) their direct children. So, the time complexity is $3k \in \Theta(k)$. The following algorithm also works, but its time complexity is $\Theta(k \log n)$ (gets partial mark): Keep removing the smallest element (extract min) until the root is larger than c. Report all extracted elements.

   (b) No, the result is not necessary a heap. Consider the following heap: 1 2 20 3 4 30 40. Changing elements $A[1]$ and $A[2]$ (i.e., $i = 0$) results in an array which is not a heap (20 is larger than its children 3 and 4).

   (c) First, replace $A[i]$ with $A[n-1]$. Next, bubble up on $A[i]$ and then bubble down on $A[i]$ (its new position). Note that bubble up is required. To see that consider the following heap $1, 100, 2, 200, 30, 3, 4$ where key 200 is deleted.