CSC238:Discrete Mathematics for Computer Science – Spring 2003

Solutions for Assignment 2

1. Prove that the `Mod(`x`, `m`)` program defined below is correct.

   `Mod(`x`, `m`)`
   Precondition: x, m are natural numbers, $m > 0$.
   Postcondition: `Mod(`x`, `m`)` returns a natural number r, with $0 \leq r < m$, such that there exists an integer n with $x = nm + r$.

```
1    r := x
2    while r ≥ m do
3        r := r − m
4    end while
5    return r
```

A loop invariant is **L:** "$x = nm + r$ for some natural number n". In order to write the proof it is convenient to explicitly refer to the number of iterations the loop has performed, say i, and let r_i denote the values of variables r **after** iteration i. (We use $i = 0$ to denote the state of the variables just before the loop begins execution for the first time.) That is, we define:

> $L(i)$: If the loop body has executed i times, then $x = im + r_i$ with r_i a natural number.

Lemma 1. If x, m satisfy the precondition then $L(i)$ is true for each natural number i.

Proof of Lemma 1. Suppose x and m satisfy the precondition. We will use mathematical induction to prove $L(i)$ is satisfied for each natural number i.

Base Case: $i = 0$. From line 1 of the program we have $r_0 = x$. Therefore, the first time the loop is reached (at line 2) we have $x = 0m + r_0$, as desired. Also, by the precondition, x is a natural number, therefore r_0 is a natural number. Therefore $L(0)$ is true.

Let $i \geq 0$ be a natural number.

Induction Hypothesis. Suppose $L(i)$ is true.

Induction Step. We need to prove $L(i + 1)$ is true. There are three cases.

Case 0. The loop may have terminated before completing i iterations, in which case $L(i + 1)$ is trivially true (i.e. the "if" clause is not satisfied).

Case 1. The loop has executed exactly i times and $r_i < m$. From line 2 we see the loop terminates. Therefore the loop never completes $i+1$ iterations, and $L(i+1)$ is trivially true.

Case 2. The loop has executed exactly i times and $r_i \geq m$. Then line 3 is executed, and

$$r_{i+1} = r_i - m. \tag{1}$$

Since $r_i \geq m$ it follows that $r_{i+1} \geq 0$. Also, since r_i is a natural number (by the induction hypothesis), and m is a natural number (by the precondition), we have r_{i+1} is a natural number. Furthermore, we see by the induction hypothesis $L(i)$ that

$$\begin{aligned} x &= im + r_i \\ &= im + r_{i+1} + m, \quad \text{by equation (1)}, \\ &= (i+1)m + r_{i+1}. \end{aligned}$$

Therefore $L(i+1)$ is true.

Since these cases cover all the possibilities, we conclude that $L(i+1)$ must be true.

It follows from mathematical induction that $L(i)$ is true for all natural numbers $i \geq 0$. This completes the proof of Lemma 1.

We can use Lemma 1 to prove **partial correctness**. Indeed, if the precondition holds and if the loop terminates after i steps then, from Lemma 1, $L(i)$ must be true. Also, since the loop terminates on this iteration, the loop condition $r_i \geq m$ must fail to hold. That is, $r_i < m$. But from $L(i)$ we know r_i must be a natural number, so $r_i \geq 0$. Also, from $L(i)$ we find that $x = im + r_i$. Therefore the postcondition holds with $n = i$.

Finally, we need to prove **termination** (i.e. the program terminates after a finite number of steps). Define the sequence $< r_0, r_1, r_2, \ldots >$ consisting of one term r_i for each iteration i which the algorithm performs (along with r_0 for the initial condition). From the precondition, the loop invariant, Lemma 1, and equation (1), it follows that the sequence $< r_0, r_1, r_2, \ldots >$ is a decreasing sequence of natural numbers. Hence it must be finite. This proves termination.

---

2. Use a loop invariant to prove that the following program is correct.

   Precondition: x is natural number.
   Postcondition: $y = 0$.

   ```
   y := x * x
   while y ≠ 0 do
       x := x − 1
   ```

```
4        y := y - 2 * x - 1
5    end while
```

We use x_i and y_i to denote the values of variables x and y after the i^{th} iteration. Consider the loop invariant:

> $L(i)$: If the loop body has executed i times, then x_i is a natural number and $y_i = x_i^2$.

Lemma 2. If x satisfies the precondition then $L(i)$ is true for each natural number i.

Proof of Lemma 2. Suppose x satisfies the precondition, i.e. x is a natural number. We will use mathematical induction to prove $L(i)$ is satisfied for each natural number i.

Base Case: $i = 0$. When the loop at line 2 of the program is first reached we have, from line 1, that $x_0 = x$ and $y_0 = x_0^2$. By the precondition x is a natural number, and so x_0 is too. Thus $L(0)$ holds.

Let $i \geq 0$ be a natural number.

Induction Hypothesis. Suppose $L(i)$ is true.

Induction Step. We need to prove $L(i+1)$ is true. There are three cases.

Case 0. The loop may have terminated before completing i iterations, in which case $L(i+1)$ is trivially true.

Case 1. The loop has executed exactly i times and $x_i = 0$. By the induction hypothesis, $y_i = x_i^2$, so $y_i = 0$. Therefore the loop terminates on line 2, never completes $i{+}1$ iterations. Thus $L(i+1)$ is trivially true.

Case 2. The only remaining case is that the loop has executed exactly i times and $x_i > 0$. (Note that x_i cannot be negative since the induction hypothesis $L(i)$ ensures that x_i is a natural number.) By the induction hypothesis, $y_i = x_i^2$, and so $y_i > 0$. Therefore lines 3 and 4 are executed, and cause the variables x and y to be updated to x_{i+1} and y_{i+1}. Just before line 5 is executed we have

$$x_{i+1} = x_i - 1. \tag{2}$$

Since $x_i > 0$ we know $x_{i+1} \geq 0$ is a natural number. Also

$$\begin{aligned} y_{i+1} =& y_i - 2x_{i+1} - 1, \\ =& x_i^2 - 2(x_i - 1) - 1, \text{ by } L(i) \text{ and equation (2) ,} \\ =& x_i^2 - 2x_i + 1 = (x_i - 1)^2, \text{ by algebra.} \\ =& x_{i+1}^2, \text{ by equation (2) again.} \end{aligned}$$

Therefore we have proven that $L(i+1)$ must be true.

Since these cases cover all the possibilities, we conclude that $L(i+1)$ must be true.

Lemma 2 now follows by mathematical induction.

To prove **partial correctness** note that if the loop terminates it must be the case that $y = 0$. Therefore the postcondition must be satisfied. (Note, we did not need Lemma 2 here!)

To prove **termination** note that, if the precondition is satisfied, then from Lemma 2, equation (2), and the loop invariant $L(i)$, we have $< x_0, x_1, x_2, \ldots >$ is a decreasing sequence of natural numbers. Hence this sequence must be finite. Therefore the loop must terminate.

---

3. Prove that the loop below terminates if the precondition is satisfied before the loop starts.

Precondition: x, y are natural numbers, and x is even.

```
1    while x ≠ 0 do
2        if y ≥ 1 then
3            y := y − 3
4            x := x + 2
5        else
6            x := x − 2
7        end if
8    end while
```

Using x_i and y_i to denote the values of variables x and y after the i^{th} iteration, consider the loop invariant:

> $L(i)$: If the loop body has executed i times, then x_i is an even natural number and $y_i \geq -2$ is an integer.

Lemma 3. If x, y satisfy the precondition then $L(i)$ is true for each natural number i.

Proof of Lemma 3. Suppose x and y satisfy the precondition, that is, they are both natural numbers and x is even. We will use mathematical induction to prove $L(i)$ is satisfied for each natural number i.

Base Case: $i = 0$. When the program starts and the loop in line 1 is reached for the first time we have, from the precondition, that $x_0 \geq 0$, x_0 is even, and $y_0 \geq 0$ is an integer. Thus $L(0)$ holds.

Let $i \geq 0$ be a natural number.

Induction Hypothesis. Suppose $L(i)$ is true.

Induction Step. We need to prove $L(i+1)$ is true. There are three cases.

Case 0. The loop may have terminated before completing i iterations, in which case $L(i+1)$ is trivially true.

Case 1. The loop has executed exactly i times and $x_i = 0$. Therefore the loop terminates and never completes $i+1$ iterations, and $L(i+1)$ is trivially true.

Case 2. The only remaining case is that the loop has executed exactly i times and $x_i \neq 0$. Note that the induction hypothesis ensures that $x_i \geq 0$ is an even natural number. It follows that $x_i \geq 2$. Therefore the loop body is executed for the $(i+1)^{st}$ time.

By the induction hypothesis $y_i \geq -2$ must be an integer. There are two subcases.

Case 2a. If $y_i \geq 1$ then lines 3 and 4 are executed and it follows that

$$y_{i+1} = y_i - 3, \quad \text{and } x_{i+1} = x_i + 2. \tag{3}$$

Therefore $y_{i+1} \geq -2$ is an integer. Moreover, since we established above that $x_i \geq 2$ and x_i is even, we find from (3) that $x_{i+1} \geq 4$ and x_{i+1} must also be even. Therefore $L(i+1)$ must be true in this subcase.

Case 2b. Otherwise $y_i < 1$. In this case line 6 is executed, and it follows that

$$y_{i+1} = y_i, \quad \text{and } x_{i+1} = x_i - 2. \tag{4}$$

Since we established above that $x_i \geq 2$ and x_i is even, we find from (4) that $x_{i+1} \geq 0$ and x_{i+1} must also be even. Clearly, from $L(i)$ and $y_{i+1} = y_i$, we have $y_{i+1} \geq -2$ is an integer. Therefore $L(i+1)$ must also be true in this subcase.

Since these cases cover all the possibilities, we conclude that $L(i+1)$ must be true.

It follows from mathematical induction that $L(i)$ is true for all natural numbers $i \geq 0$. This completes the proof of Lemma 3.

In order to prove **termination**, suppose x and y satisfy the precondition. Let $i \geq 0$ be a natural number and suppose the program has executed the loop i times. Define $p_i = x_i + y_i + 2$. Then, from Lemma 3, the loop invariant $L(i)$ is true. It follows that $p_i = x_i + y_i + 2$ is an integer. Furthermore, the loop invariant ensures $x_i \geq 0$ and $y_i \geq -2$, so we have $p_i \geq 0 + (-2) + 2 = 0$. Therefore p_i is a natural number. Moreover, if the loop body executes at least one more time, namely the $(i+1)^{st}$ time, then a similar argument shows that p_{i+1} is a natural number.

Finally, from the program it is clear that the loop body causes the variables x and y to be updated according to either equation (3) or equation (4). Therefore, we have

$$\begin{aligned} p_{i+1} =& x_{i+1} + y_{i+1} + 2, \\ \leq& \max\{(y_i - 3) + (x_i + 2), y_i + (x_i - 2)\} + 2, \quad \text{from equations (3, 4)}, \\ =& (y_i + x_i - 1) + 2, \quad \text{by algebra}, \\ =& p_i - 1, \quad \text{by algebra and the definition of } p_i. \end{aligned}$$

Therefore it follows that the sequence $< p_0, p_1, p_2, \ldots >$ is a decreasing sequence of natural numbers. Hence it must be finite. This proves the loop terminates after a finite number of steps.

---

4. Consider the following precondition, postcondition pair:

   `BeLow(`A`, `n`, `y`)`
   Precondition: $n \geq 1$ and A is an array of length n sorted in non-decreasing order. That is, for any integer i with $1 \leq i < n$, we have $A[i] \leq A[i+1]$. Finally, y is an integer such that $y \leq A[n]$.
   Postcondition: `BeLow(`A`, `n`, `y`)` returns the minimum integer $k \geq 1$ such that $A[k] \geq y$.

   (a) Write a binary search style algorithm for solving this problem. That is, your algorithm **must** run in $O(\log(n))$ time, like binary search does. Algorithms which require time $\Omega(n)$ will receive a mark of 0. (Hint: First write a loop invariant for your program, and then write the loop body.)

   (b) Write a loop invariant for your program and use it to prove that your program is correct.

   (c) Prove that your program executes in $O(k)$ steps when the length of A is $n = 2^k$.

Solution 4a. We will maintain two indices f and l such that the answer lies between f and l. In particular, suppose the following loop invariant is satisfied:

> $L(i)$: If the loop body has executed i times, then f_i, l_i are natural numbers such that $0 \leq f_i < l_i \leq n = \text{length}(A)$, with $A[l_i] \geq y$ and if $f_i > 0$ then $A[f_i] < y$.

With this loop invariant in mind, we can write the program.

```
BeLow(A, n, y)
```

Precondition: $n \geq 1$ and A is an array of length n sorted in non-decreasing order. That is, for any integer i with $1 \leq i < n$, we have $A[i] \leq A[i{+}1]$. Finally, y is an integer such that $y \leq A[n]$.

Postcondition: `BeLow(`A`, `n`, `y`)` returns the minimum integer $k \geq 1$ such that $A[k] \geq y$.

```
1    f := 0
2    l := n
3    while f + 1 < l do
4        m := (f + l) div 2
5        if A[m] >= y then
6            l := m
7        else
8            f := m
9        end if
10   end while
11   return l
```

Solution 4b. We will use the loop invariant in part (a) to prove the program is correct.

Lemma 4.1. Suppose A, n, and y satisfy the preconditions. Then for each natural number i the loop invariant $L(i)$ is true.

Proof of Lemma 4.1. Suppose A, n and y satisfy the precondition. We will use mathematical induction to prove $L(i)$ is satisfied for each natural number i.

Base Case: $i = 0$. From the precondition we have $A[n] \geq y$. From lines 1 and 2 we see $f_0 = 0$ and $l_0 = n$. Therefore it follows that $A[l_0] \geq y$. Since $n \geq 1$ by the precondition we have $0 = f_0 < l_0 = n$. Thus $L(0)$ holds.

Let $i \geq 0$ be a natural number.

Induction Hypothesis. Suppose $L(i)$ is true.

Induction Step. We need to prove $L(i+1)$ is true. There are three cases.

Case 0. The loop may have terminated before completing i iterations, in which case $L(i+1)$ is trivially true.

Case 1. The loop has executed exactly i times and $f_i + 1 \geq l_i$. By the condition on line 3 the loop terminates and never completes $i+1$ iterations. Thus $L(i+1)$ is trivially true.

Case 2. The only remaining case is that the loop has executed exactly i times and $f_i + 1 < l_i$. In this case the loop body is executed.

On line 5 the value of the variable m is set to $m_i = (f_i + l_i) \text{ div } 2$. Since $f_i + 1 < l_i$ and f_i, l_i are both natural numbers (by the induction hypothesis $L(i)$), we have $l_i \geq f_i + 2$. Therefore m_i satisfies

$$\begin{aligned} f_i + 1 &= (f_i + f_i + 2) \text{ div } 2, \quad \text{by algebra,} \\ &\leq (f_i + l_i) \text{ div } 2, \quad \text{since } f_i + 2 \leq l_i, \\ &= m_i, \quad \text{by defn of } m_i, \\ &= (f_i + l_i) \text{ div } 2, \quad \text{by defn of } m_i, \\ &\leq (l_i - 2 + l_i) \text{ div } 2, \quad \text{since } f_i \leq l_i - 2, \\ &= l_i - 1, \quad \text{by algebra.} \end{aligned}$$

Therefore m_i is a natural number, and

$$0 < f_i + 1 \leq m_i \leq l_i - 1 < n. \tag{5}$$

The inequalities on the far left and right in (5) above arise from $f_i \geq 0$ and $l_i \leq n$, which in turn follow from the induction hypothesis $L(i)$.

After computing m_i on line 4, the program continues execution on line 5. Notice that from (5) we know the array index m_i is within bounds for the array access $A[m_i]$. There are two cases for line 5:

Case 2a. $A[m_i] \geq y$. Then line 6 is executed, and it follows that

$$f_{i+1} = f_i \text{ and } l_{i+1} = m_i. \tag{6}$$

Using inequality (5) and the induction hypothesis $L(i)$ it now follows that $L(i+1)$ must be satisfied.

Case 2b. $A[m_i] < y$. Then line 8 is executed, and it follows that

$$f_{i+1} = m_i \text{ and } l_{i+1} = l_i. \tag{7}$$

Using inequality (5) along with the induction hypothesis $L(i)$ it again follows that $L(i+1)$ must be satisfied.

Since these cases cover all the possibilities, we conclude that $L(i+1)$ must be true.

It follows from mathematical induction that $L(i)$ is true for all natural numbers $i \geq 0$. This completes the proof of Lemma 4.1.

To prove **partial correctness** of the program in part 4a note that, if the loop terminates on the i^{th} iteration, it must be the case that $f_i + 1 \geq l_i$ (since this is the only way the loop condition on line 3 fails). By Lemma 4.1 and the loop invariant $L(i)$ we find that $f_i + 1 \leq l_i$. Therefore, on termination we must have

$$f_i + 1 = l_i. \tag{8}$$

There are now two cases to consider:

Case 1. Suppose $f_i = 0$. Then, by (8), $l_i = f_i + 1 = 1$ and, by the loop invariant $L(i)$, we have $A[l_i] \geq y$. Thus $k = l_i = 1$ is the smallest index such that $A[k] \geq y$, and the value $l_i = 1$ returned on line 11 is correct (i.e. the postcondition is satisfied).

Case 2. Alternatively, $f_i \neq 0$. Since the loop invariant ensures f_i is a natural number we must have $f_i > 0$. Moreover, from $L(i)$ we also have $f_i < n$ and $A[f_i] < y$. In addition, from $L(i)$ and (8) it follows that $A[f_i + 1] = A[l_i] \geq y$. Therefore $k = l_i$ is the smallest index at which $A[k] \geq y$. And therefore the postcondition is again satisfied.

Since these are the only two possible cases for f_i on termination, we have proven partial correctness.

To prove **termination** assume the precondition is satisfied. Consider the sequence $< d_0, d_1, \ldots >$, where $d_i = l_i - f_i$ is defined only if the loop executes at least i times. Note that, if the loop executes i times, then Lemma 4.1 and the loop invariant $L(i)$ ensure that d_i is a natural number. If the loop executes at least $i+1$ times then a similar argument shows that d_{i+1} is also a natural number. Moreover, we have

$$\begin{aligned} 0 &< d_{i+1}, \quad \text{by Lemma 4.1 and the loop invariant } L(i+1), \\ &\leq \max\{l_i - m_i, m_i - f_i\}, \quad \text{since equation (6) or (7) must apply,} \\ &\leq \max\{l_i - (f_i + 1), (l_i - 1) - f_i\}, \quad \text{by (5),} \\ &= d_i - 1, \text{ by algebra.} \end{aligned}$$

Therefore $< d_0, d_1, d_2, \ldots >$ is a decreasing sequence of natural numbers. Hence this sequence must be finite. Therefore the loop in `BeLow` must terminate, and hence `BeLow` itself must terminate.

Solution 4c. Consider the predicate:

$P(i)$: The loop body executes at least i times, and $l_i = f_i + 2^{k-i}$.

Lemma 4.2 If the A, n, and y satisfy the preconditions for `BeLow` and the length of A is $n = 2^k$ for some integer $k \geq 0$, then $P(i)$ is true for $i = 0, 1, \ldots, k$.

Before proving Lemma 4.2 we will use it to prove that `Below` executes in $O(k)$ steps. Here we take a step to be a simple operation such as an assignment, an arithmetic operation, a comparison opertation, an array reference, or the return statement. Therefore, it takes a bounded number of steps to execute any individual line in the program once. In particular, the initialization (lines 1 and 2), the loop body (lines 4 through 9), and the return statement, each execute in a bounded number of steps. Therefore each of these blocks of code execute in $O(1)$ steps.

From Lemma 4.2 we find that $l_i - f_i = 2^{k-i}$. Therefore $l_i - f_i > 1$ for $i < k$ and $l_k - f_k = 1$. Thus, from $P(i)$ and the loop condition on line 3, we see that the loop must execute exactly k times, exiting when $i = k$.

Since each iteration of the loop requires $O(1)$ steps, we can use the rule of products to conclude that all k iterations of the loop require a total of $kO(1) = O(k)$ steps. Since the initialization and return statements each require $O(1)$ steps, we find the total number of steps for `Below` is $O(1) + O(k) + O(1)$. By the rule of sums, this is just $O(k)$. Therefore the overall number of steps for `Below` is $O(k)$, as desired.

Proof of Lemma 4.2 Let A, n, and y satisfy the preconditions for `BeLow` and suppose the length of A is $n = 2^k$ for some integer $k \geq 0$. We will use induction to prove $P(i)$ is true for $0 \leq i \leq k$.

Base Case: $i = 0$. From lines 1 and 2 we see $f_0 = 0$ and $l_0 = n = 2^k$. Therefore $l_0 - f_0 = 2^{k-0}$. Clearly the loop must execute at least 0 times. Thus $P(0)$ holds.

Let $0 \leq i < k$ be a natural number.

Induction Hypothesis. Suppose $P(i)$ is true.

Induction Step. We need to prove $P(i+1)$ is true.

By the induction hypothesis $P(i)$ the loop executes at least i times, resulting in $l_i = f_i + 2^{k-i}$. Since $i < k$ we have $k - i \geq 1$ and $l_i = f_i + 2^{k-i} \geq f_i + 2^1$. Therefore the loop condition on line 3 is satisfied, and it follows that the loop body is executed at least one more time, namely the $(i+1)^{st}$ time. Immediately after line 4 during this execution of the loop body we have

$$\begin{aligned} m_i &= (f_i + l_i) \text{ div } 2, \\ &= (f_i + f_i + 2^{k-i}) \text{ div } 2 \ \text{ since } l_i = f_i + 2^{k-i}, \\ &= f_i + 2^{k-i-1} \ \text{ since } k - i \geq 1. \end{aligned} \tag{9}$$

As in the proof of Lemma 4.1, depending on whether or not $A[m_i] \geq y$, we have either (6) or (7). If line 6 is executed then equation (6) is satisfied and $l_{i+1} - f_{i+1} = m_i - f_i = 2^{k-i-1}$. Here we have used (9). Alternatively, line 8 is executed and equation (7) must be satisfied. In this case $l_{i+1} - f_{i+1} = l_i - m_i$. By (9) and the induction hypothesis $P(i)$, we have $l_i - m_i = (f_i + 2^{k-i}) - (f_i + 2^{k-i-1}) = 2^{k-i} - 2^{k-i-1} = 2^{k-i-1}$. Therefore, in both cases, we have shown that $l_{i+1} - f_{i+1} = 2^{k-(i+1)}$. Since these are the only two possible cases, we conclude that the loop executes at least $i+1$ times and $l_{i+1} = f_{i+1} + 2^{k-(i+1)}$. That is, $P(i+1)$ must be satisfied.

Therefore it follows by mathematical induction that $P(i)$ is true for each i such that $0 \leq i \leq k$. This completes the proof of Lemma 4.2.

5. Show that for every real number $d > 0$, the function $f_d(n) = \sum_{k=0}^{n} k^d$ satisfies $f_d(n) \in \Theta(n^{d+1})$.

Let $d > 0$ be a real number.

First we will prove that $f_d(n) \in O(n^{d+1})$. In particular,

$$\begin{aligned}
f_d(n) &= \sum_{k=0}^{n} k^d \\
&\leq \sum_{k=0}^{n} n^d, \quad \text{since } d > 0 \text{ and } 0 \leq k \leq n \text{ imply } k^d \leq n^d, \\
&= (n+1)n^d, \quad \text{by algebra,} \\
&\leq (n+n)n^d \quad \text{so long as } n \geq 1, \\
&= 2n^{d+1}, \quad \text{by algebra.}
\end{aligned}$$

Therefore $f_d(n) \leq 2n^{d+1}$ for all $n \geq 1$ and thus $f_d(n) \in O(n^{d+1})$.

Next we will prove that $f_d(n) \in \Omega(n^{d+1})$. In particular,

$$\begin{aligned}
f_d(n) &= \sum_{k=0}^{n} k^d \geq \sum_{k=\lceil n/2 \rceil}^{n} k^d, \\
&\geq \sum_{k=\lceil n/2 \rceil}^{n} (n/2)^d, \quad \text{since } d > 0 \text{ and each } k \geq \lceil n/2 \rceil \geq n/2 \ , \\
&= (n - \lceil n/2 \rceil + 1)(n/2)^d, \\
&\geq (n - (n/2 + 1/2) + 1)(n/2)^d, \quad \text{since } \lceil n/2 \rceil \leq n/2 + 1/2 \ , \\
&= (n/2 + 1/2)(n/2)^d \geq (n/2 + 0)(n/2)^d, \\
&= (n/2)^{d+1}.
\end{aligned}$$

Therefore $f_d(n) \geq (n/2)^{d+1}$ for all $n \geq 0$ and thus $f_d(n) \in \Omega(n^{d+1})$.

Since $f_d(n) \in O(n^{d+1})$ and $f_d(n) \in \Omega(n^{d+1})$, it follows from the definition of $\Theta()$ that $f_d(n) \in \Theta(n^{d+1})$.

---

6. Let k be an integer and let $f(n)$ and $g(n)$ be positive, real-valued functions defined for natural numbers $n \geq k$ (that is, $f(n), g(n) > 0$ for all $n \geq k$). Suppose $\lim_{n\to\infty} f(n)/g(n) = x$ for some real number $x < \infty$.

(a) Prove that $f(n) \in O(g(n))$.

(b) Prove that if $x \neq 0$ then $f(n) \in \Theta(g(n))$.

(c) Prove that if $x = 0$ then $g(n) \notin O(f(n))$.

(d) Prove $\log(\log(n)) \in O(\log(n))$ but $\log(\log(n)) \notin \Theta(\log(n))$ using parts (a-c) above.

(e) Does the limit $\lim_{n\to\infty} f(n)/g(n)$ exist whenever $f(n) \in \Theta(g(n))$? Explain.

Solution 6a. Let $\epsilon > 0$. By the definition of limit, if $\lim_{n\to\infty} f(n)/g(n) = x < \infty$, then for there exists a $N_\epsilon > 0$ such that $|f(n)/g(n) - x| < \epsilon$ for all $n \geq N_\epsilon$. Therefore it must be the case that $f(n)/g(n) - x < \epsilon$ for all $n \geq N_\epsilon$. That is, $f(n) < (x + \epsilon)g(n)$ for all $n \geq \max\{N_\epsilon, k\}$ (here we have used $g(n) > 0$ for $n \geq k$). Therefore we have shown that $f(n) \in O(g(n))$.

Solution 6b. Since $f(n)/g(n) > 0$ for $n \geq k$ the limit $\lim_{n\to\infty} f(n)/g(n) = x$ must be in the closure of the positive real line, that is $x \geq 0$. Therefore, if $x \neq 0$ then it must be the case that $x > 0$. Let $\epsilon = x/2$. Then $\epsilon > 0$ and, by the definition of a limit, there must exist an N_ϵ such that $|f(n)/g(n) - x| < \epsilon$ for all $n \geq N_\epsilon$. Therefore it follows that $x - f(n)/g(n) < \epsilon$ for all $n \geq N_\epsilon$. And finally, $f(n) > (x - \epsilon)g(n)$ for all $n \geq \max\{k, N_\epsilon\}$. Notice that $x - \epsilon = x - x/2 = x/2 > 0$. Therefore it follows that $f(n) \in \Omega(g(n))$. Since, by part 6a, $f(n)$ is also in $O(g(n))$ we conclude $f(n) \in \Theta(g(n))$.

Solution 6c. Let $x = 0$. We will prove $g(n) \notin O(f(n))$ by contradiction.

Suppose $g(n) \in O(f(n))$. By the definition of $O()$ there exist constants N, c such that $g(n) \leq cf(n)$ for all $n \geq N$. Without loss of generality we can assume $c > 0$ and $N \geq k$ (since otherwise we can replace c and N by $c + 1$ and $\max\{N, k\}$). Therefore $1/c \leq f(n)/g(n)$ for all $n \geq N$ (here we have used $g(n) > 0$ for $n \geq N \geq k$). However $\lim_{n\to\infty} f(n)/g(n) = x = 0$ implies that, for $\epsilon = 1/(2c)$, there exists an N_ϵ such that $f(n)/g(n) \leq 1/(2c) < 1/c$ for all $n \geq N_\epsilon$. We therefore have the contradiction that $f(n)/g(n) < 1/c \leq f(n)/g(n)$ for $n = \max\{N, N_\epsilon\}$. Therefore $g(n) \notin O(f(n))$.

Solution 6d. Let $f(y) = \log(\log(y))$ and $g(y) = \log(y)$. Notice that both $f(y)$ and $g(y)$ are increasing functions of y and $\lim_{y\to\infty} \log(y) = \lim_{y\to\infty} \log(\log(y)) = \infty$. Therefore, by l'Hospital's rule, we find

$$\lim_{y\to\infty} \frac{\log(\log(y))}{\log(y)} = \lim_{y\to\infty} \frac{\frac{d}{dy}\log(\log(y))}{\frac{d}{dy}\log(y))} = \lim_{y\to\infty} \frac{(1/(y\log(y)))}{(1/y)} = \lim_{y\to\infty} \frac{1}{\log(y)} = 0.$$

Therefore, $\lim_{n\to\infty} f(n)/g(n) = 0$, and parts (6a) and (6c) apply. That is, $f(n) \in O(g(n))$ but $g(n) \notin O(f(n))$. Recall that if $f(n) \in \Theta(g(n))$ then $g(n) \in O(f(n))$. Therefore, from $g(n) \notin O(f(n))$ we can conclude that it $f(n) \notin \Theta(g(n))$, as desired.

Solution 6e. No, the limit need not exist.

Consider $g(n) = 1$ and $f(n) = 2 + (-1)^n$. Then, clearly, $g(n) \leq f(n) \leq 3g(n)$ for all $n \geq 0$. Therefore $f(n) \in \Theta(g(n))$. However, notice that $f(n)/g(n) = 2 + (-1)^n$ takes on values 3 (when n is even) and 1 (when n is odd).

Suppose the limit exists and $\lim_{n\to\infty} f(n)/g(n) = x$. Consider $\epsilon = 1/2$, then there must be an N such that $|x - f(n)/g(n)| < \epsilon$ for all $n \geq N$. However, this implies that $|x-1| < \epsilon$ and $|x - 3| < \epsilon$. Therefore $x < 1 + \epsilon = 1.5$, and $x > 3 - \epsilon = 2.5$. This is a contradiction, and hence $\lim_{n\to\infty} f(n)/g(n)$ does not exist.
