Sample Solutions to Homework #1

1. (Section 3.3, Exercise 26, page 254) Let $g(n)$ be the number of cardinality-2 subsets of a set with $n$ elements. By Exercise 25, we have $g(n) = n(n-1)/2$. We will use $g(n)$ later in our proof. Let $f(n)$ be the number of cardinality-3 subsets of a set with $n$ elements. We need to prove that for all $n \geq 3$, $f(n) = n(n-1)(n-2)/6$. The base case is $n = 3$. When $n = 3$, there is exactly 1 subset (the set itself) that has three elements, and $n(n-1)(n-2)/6 = 3 \cdot 2 \cdot 1/6 = 1$. Hence, the claim is true for the base case. Assume the inductive hypothesis, namely, for a pile of $n$ elements, we have $f(n) = n(n-1)(n-2)/6$. We then need to prove that $f(n+1) = (n+1)n(n-1)/6$. Fix an arbitrary element $a$ in $S$ and let $T = S - \{a\}$ (hence $|T| = n$). Let $x$ be the number of cardinality-3 subsets that contains $a$, and let $y$ be the number of cardinality-3 subsets that does not contain $a$. Then $x = g(n)$ (because the rest of the two elements in the subset can only be chosen from $T$), and $y = f(n)$ (because all the three elements are chosen from $T$). Therefore,

$$f(n+1) = \{\text{definition of } x \text{ and } y\}$$

$$= x + y$$

$$= \{\text{reasons stated above}\}$$

$$g(n) + f(n)$$

$$= \{\text{Exercise 25 and the inductive hypothesis}\}$$

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

$$= \{\text{arithmetic}\}$$

$$\frac{(n+1)n(n-1)}{6}.$$ 

This completes the inductive step and the whole proof.

2. (Section 3.3, Exercise 40, page 254) We use strong induction. In the base case, $n = 1$ and no split is performed. So the sum is 0, which equals $n(n-1)/2$. Therefore the claim is true for the base case. Assume the strong inductive hypothesis, namely, for a pile of $i$ stones, where $1 \leq i \leq n$, the sum computed is $i(i-1)/2$. Now we need to prove that for a pile of $n+1$ stones, the sum computed is $(n+1)n/2$. Suppose that the player first splits the stones into two piles of $r$ and $n+1-r$ stones. We do not know the exact value of $r$ (that is up to the player and that is why we need strong induction), but we do know that $1 \leq r \leq n$ and $1 \leq n+1-r \leq n$ (otherwise it is not a split). Hence, the first splitting gives us a product of $r(n+1-r)$. The rest of the sum is from splitting the two smaller piles of stones. By the inductive hypothesis, these two sums are $r(r-1)/2$ and $(n+1-r)(n-r)/2$. The final sum therefore is

$$r(n+1-r) + \frac{r(r-1)}{2} + \frac{(n+1-r)(n-r)}{2}$$

$$= \{\text{arithmetic}\}$$

$$\frac{(n+1)n}{2},$$

no matter what $r$ is. This completes our inductive step and the whole proof.

3. (Section 3.3, Exercise 52, page 255) The mistake is in applying the inductive hypothesis to $\max(x-1, y-1)$. Although $x$ and $y$ are positive integers, $x-1$ and $y-1$ need not be (e.g., when $x = 1$ and $y = 1$). Therefore, the inductive hypothesis cannot be applied.

4. (Section 3.3, Exercise 62, page 256) The base case is $n = 1$. When $n = 1$, $4^{n+1} + 5^{2n-1} = 4^{1+1} + 5^{2-1} = 21$, which is divisible by 21. Assume that $4^{n+1} + 5^{2n-1}$ is divisible by 21. We then need to prove that
$4^{(n+1)+1} + 5^{2(n+1)-1} = 4^{n+2} + 5^{2n+1}$ is also divisible by 21. By (somewhat tricky) arithmetic manipulation, we have

$$4^{n+2} + 5^{2n+1} = 4 \cdot 4^{n+1} + 25 \cdot 5^{2n-1} = 4(4^{n+1} + 5^{2n-1}) + 21 \cdot 5^{2n-1},$$

which is also divisible by 21: the first term is divisible by 21 due to the inductive hypothesis and the second term is clearly divisible by 21. This completes the inductive step and the whole proof. (Remark: The above arithmetic manipulation will become less tricky once you have seen several of such exercises.)

5. (Section 3.3, Exercise 66, page 256) (a) Let $s = 1$ and $t = 1$, then $a + b \in S$. Therefore, $S$ is non-empty. (b) The Well-Ordering Property states that every non-empty set of positive integers have a least element. Part (a) has established that $S$ is non-empty. Therefore, there is a least element in $S$. Let this element be $c$, and let $c = as_0 + bt_0$ for some integers $s_0$ and $t_0$. (c) Since $d$ is a common divisor of $a$ and $b$, $d | a$ and $d | b$. Therefore, $d | as + bt$ for all integers $s$ and $t$. Thus, $d | c$, because $c = as_0 + bt_0$. (d) We prove this by contradiction. Suppose $c \nmid a$, then by arithmetic, $a$ can be written as $a = qc + r$, where $0 < r < c$. Hence $r = a - qc = a - q(as_0 + bt_0) = a(1 - qs_0) + b(-qt_0)$, namely, $r$ satisfies the definition of $S$. Therefore, $r \in S$. Furthermore, $0 < r < c$. But this contradicts that $c$ is the least element in $S$. Therefore, our assumption that $c \nmid a$ is false. Hence, $c | a$. By the same argument, $c | b$. (e) The number $c$ as defined in part (b) is the greatest common divisor of $a$ and $b$. Part (d) has established that $c$ is a common divisor of $a$ and $b$, and part (c) has established that every common divisor of $a$ and $b$ is a divisor of $c$. Hence, $c$ is the greatest common divisor. Greatest common divisor is unique because one cannot find two greatest common divisors, each greater than the other.

6. (Section 3.4, Exercise 6(d), page 271) Invalid, because when $n = 1$, $f(n)$ is defined in two conflicting ways: $f(1) = 1$ and then $f(1) = 2f(1 - 1) = 2f(0) = 0$.

7. (Section 3.4, Exercise 14, page 271) When $n = 1$, $f_2f_0 - f_1^2 = 1 \cdot 0 - 1^2 = (-1)^1$. The base case is hence established. Assume the inductive hypothesis, namely, $f_{n+1}f_{n-1} - f_n^2 = (-1)^n$. Then

$$f_{n+2}f_{n-1} - f_n^2$$

= \{definition of Fibonacci sequence\}

$$=(f_{n+1} + f_n)f_n - f_n^2$$

= \{arithmetic\}

$$= f_n^2 - f_{n+1}(f_{n+1} - f_n)$$

= \{definition of Fibonacci sequence\}

$$= f_n^2 - f_{n+1}f_{n-1}$$

= \{inductive hypothesis\}

$$= -(-1)^n$$

= \{arithmetic\}

$$= (-1)^{n+1}$$

This completes the inductive step and the whole proof.

8. (Section 3.4, Exercise 32, page 272) (a) Let $\lambda$ denote the empty string. Our recursive definition is: $\text{ones}(\lambda) = 0$ and $\text{ones}(wx) = \text{ones}(w) + x$, where $w$ is a bit string and $x$ is a single bit (viewed as an integer when being added). (See also Example 9 on page 263.) (b) We induct on the length of $t$. The base case is $t = \lambda$, in which case we have $\text{ones}(st) = \text{ones}(s\lambda) = \text{ones}(s) = \text{ones}(s) + 0 = \text{ones}(s) + \text{ones}(\lambda) = \text{ones}(s) + \text{ones}(t)$. For the inductive step, let $t = wx$, where $w$ is a bit string and $x$ is a single bit. Then we have
ones(st) = \{ t = wx \}
ones(s(wx)) = \{ \text{associativity of string concatenation} \}
ones((sw)x) = \{ \text{recursive definition of ones from part (a)} \}
ones(sw) + x = \{ \text{inductive hypothesis on w} \}
ones(s) + ones(w) + x = \{ \text{recursive definition of ones} \}
ones(s) + ones(wx) = \{ t = wx \}
ones(s) + ones(t).

This completes the inductive step and the proof. (See also Example 14 on page 268.)

9. (Section 3.4, Exercise 36, page 272) We induct on the length of $w_2$. The base case is $w_2 = \lambda$, in which case $(w_1w_2)^R = (w_1\lambda)^R = w_1^R = \lambda^Rw_1^R = w_2^Rw_1^R$. For the inductive step, let $w_2 = w_3x$, where $w_3$ is a string and $x$ is a symbol (the last symbol of $w_2$). Then we have

$$(w_1w_2)^R = \{ w_2 = w_3x \}
(w_1w_3x)^R = \{ \text{associativity of string concatenation} \}
((w_1w_3)x)^R = \{ \text{definition of string reversal, see Exercise 35} \}
x(w_1w_3)^R = \{ \text{inductive hypothesis} \}
x(w_3x)^Rw_1^R = \{ \text{associativity of string concatenation} \}
(xw_3^R)^Rw_1^R = \{ \text{definition of string reversal} \}
w_3x)^Rw_1^R = \{ w_3 = w_3x \}
w_2^Rw_1^R = \{ t = wx \}

This completes our inductive step and the whole proof.

10. (Section 3.4, Exercise 40, page 272) Let $S$ denote the set in question. The key observation is that if a bit string belongs to $S$, then it is the concatenation of two strings in $S$, or it is the concatenation of strings in $S$ with a 1 before, after, or in between them. Therefore, the recursive definition of $S$ is: $0 \in S$, and if $x \in S$ and $y \in S$ ($x$ and $y$ need not be different), then $xy$, $1xy$, $xy1$, $x1y \in S$. (See also Example 7 on page 262.)

11. (Section 3.4, Exercise 46, page 273) We first define the lexicographic ordering on $(m, n)$. The base case is $(m, n) = (1, 1)$, in which case $a_{m,n} = a_{1,1} = 5 = 2(1 + 1) + 1$. Hence, the base case is established. For the inductive step, assume that $a_{i,j} = 2(i + j) + 1$, for all $(i, j) < (m, n)$. Consider $a_{m,n}$. If $n = 1$, then

$$a_{m,n} = \{ \text{definition of } a_{m,n} \}
a_{m-1,n} + 2 = \{ \text{inductive hypothesis} \}
2(m - 1 + n) + 1 + 2 = \{ \text{arithmetic} \}
2(m + n) + 1,$$
which means the claim is true. If \( n > 1 \), then

\[
\begin{align*}
    a_{m,n} &= \{\text{definition of } a_{m,n}\} \\
    a_{m,n-1} + 2 &= \{\text{inductive hypothesis}\} \\
    2(m + n - 1) + 1 + 2 &= \{\text{arithmetic}\} \\
    2(m + n) + 1,
\end{align*}
\]

which also means that the claim is true. Therefore, we have completed the inductive step and the whole proof.

12. (Section 3.5, Exercise 12, page 283) The algorithm for Exercise 11 is at the back of the textbook, although it is not quite correct. In particular, the “2x” in the code should be “2”. We prove the correctness of the algorithm by using strong induction on the value of \( y \). The base case is \( y = 0 \). When \( y = 0 \), \( xy = 0 \) and the algorithm correctly returns the value. Assume that the algorithm works correctly on all smaller values of \( y \), and consider its execution on \( y \). If \( y \) is even (and thus at least 2), then the algorithm computes \( 2 \cdot \text{mult}(x, y/2) \). By the inductive hypothesis, \( \text{mult}(x, y/2) \) is correctly computed. Thus \( \text{mult}(x, y) \) is correctly computed because \( xy = 2(x \cdot y/2) \). Similarly, when \( y \) is odd, the algorithm computes \( 2 \cdot \text{mult}(x, (y-1)/2) + x \). By the inductive hypothesis, \( \text{mult}(x, (y-1)/2) \) is correctly computed. Therefore, \( \text{mult}(x, y) \) is correctly computed again because \( xy = 2(x \cdot (y-1)/2) + x \). This completes the inductive step and the whole proof.

13. (Section 3.5, Exercise 16, page 283)

```plaintext
procedure f(a, n) {
    if n = 1 then f(a, n) := a^2  
    else f(n, a) := f(a, n-1)^2
}
```

14. (Section 3.5, Exercise 18, page 283)

```plaintext
procedure g(a, n) {
    if n = 1 then g(a, n) := a  
    else if n is even then g(a, n) := g(a, n/2)^2  
    else g(a, n) := a \cdot g((n-1)/2, a)^2
}
```

(Remark: Although not immediately obvious, the above method computes \( a^n \) much more efficiently than the straightforward method of \( n - 1 \) multiplications of \( a \). For example, when \( n = 8 \). The above method computes \( a^8 \) as: \( a^8 = ((a^2)^2)^2 \) and uses only 3 multiplications, counting squaring as a multiplication. Note that \( 8 = 2^3 \), hence the name binary expansion. The straightforward method, however, uses 7 multiplications. Roughly speaking, the above method uses order \( \log n \) multiplications, while the straightforward method uses order \( n \) multiplications. The difference is huge, especially when \( n \) is large. The idea of binary expansion is widely used.)

15. (Section 3.5, Exercise 24, page 283)

```plaintext
procedure a(n) {
    if n < 3 then a(n) := n + 1  
    else a(n) := a(n-1) + a(n-2) + a(n-3)
}
```

16. (Section 3.5, Exercise 26, page 283) The iterative algorithm is much more efficient. If we use the recursive algorithm, we end up computing the small values (early terms in the sequence) over and over again. (Remark: To understand the efficiency difference better, try to write the code in your favorite language and compare their running times.)