Homework 1 Solutions

For odd-numbered problems, see solutions in the book, with exceptions noted below.

1.1 #42 (a) “Are you a liar?” doesn’t work because a truth-teller or a liar will both answer “no.”
(b) Ask “Are you a cannibal?” to get the truth-teller to say “yes,” the liar “no.”

1.3 #23 (f) \( (\forall x F(x)) \lor (\exists x \neg P(x)) \). The answer in the book is wrong.

1.3 #30 (a) Let \( D(x) \) be “\( x \) is a dog,” and let \( F(x) \) be “\( x \) has a tail.”

(d) Let \( M(x) \) be “\( x \) is a monkey” and let \( F(x) \) be “\( x \) can speak French.” The statement is \( \exists x (M(x) \land F(x)) \). Its negation is \( \forall x (M(x) \land F(x)) \), or “there is a monkey that can speak French.”

(e) Let \( P(x) \) be “\( x \) is a pig,” \( S(x) \) “\( x \) can swim,” and \( F(x) \) “\( x \) can catch fish.” The statement is \( \exists x (P(x) \land S(x) \land F(x)) \). Its negation, written without negated quantifiers, is \( \forall x (P(x) \land S(x) \land F(x)) \), or “no pig can swim and catch fish.”

1.4 #9 (j) A better answer is \( \exists x \forall y (L(x, y) \land x = y) \). The answer in the book is more accurately expressed as “there is someone who loves himself or herself and nobody else.”

2.1 #4:

```plaintext
procedure maxdiff(a_1, \ldots, a_n: integers)
    m := 0
    for i = 1, \ldots, n - 1
        if \(|a_{i+1} - a_i| > m\) then m := \(|a_{i+1} - a_i|\)
    return m := maximum absolute difference
```

If instead we wanted the maximum signed-difference \( a_{i+1} - a_i \), we would start with \( m := a_2 - a_1 \), let \( i \) go from 2 to \( n - 1 \), and drop the absolute-value signs in the “if…then” line.

2.2 #2, #23 (a) \( 17x + 11 \) is \( O(x^2) \) and not \( \Omega(x^2) \) or \( \Theta(x^2) \).
(b) \( x^2 + 1000 \) is \( \Theta(x^2) \).
(c) \( x \log x \) is \( O(x^2) \) and not \( \Omega(x^2) \) or \( \Theta(x^2) \).
(d) \( x^4/2 \) is \( \Omega(x^2) \) and not \( O(x^2) \) or \( \Theta(x^2) \).
(e) \( 2^x \) is \( \Omega(x^2) \) and not \( O(x^2) \) or \( \Theta(x^2) \).
(f) \( \lfloor x \rfloor \cdot \lfloor x \rfloor \) is \( \Theta(x^2) \).

2.2 #18: Note that \( k \) is a fixed constant, and we are considering

\[ 1^k + 2^k + \cdots + n^k \]

as a function of \( n \). There are \( n \) terms, and each term is \( O(n^k) \), so the whole sum is \( O(n)O(n^k) \), or \( O(n^{k+1}) \).

Extra problem for 2.3:
(A) The inner loop takes \( O(i) \) steps, and we always have \( i \leq n \), so the inner loop takes \( O(n) \) steps. The outer loop performs the inner loop \( O(n) \) times, for a total running time of \( O(n^2) \). This analysis is for the worst-case, when the input has no duplicates. At the opposite extreme, if it happens that \( a_1 = a_2 \), the algorithm takes constant time.
(B) A better solution is to sort the list first. After that, any duplicates will be adjacent, and we only need a linear-time scan to see if \( a_i = a_{i+1} \) for some \( i \). The sort takes \( O(n \log n) \) steps and the scan takes \( O(n) \). Since \( n \log n \) and \( n \) are both positive, and \( n \log n \) is larger, the total time is \( O(n \log n) \).