SUPPLEMENTARY EXERCISES FOR CHAPTER 3

2. a) We need to keep track of the first and second largest elements as we go along, updating as we look at the elements in the list.

```
procedure $toptwo(a_1, a_2, \ldots, a_n$ : integers)
$largest := a_1$
$second := -\infty$
for $i := 2$ to $n$
begin
        if $a_i > second$ then $second := a_i$
        if $a_i > largest$ then
        begin
                $second := largest$
                $largest := a_i$
        end
end
```

b) The loop is executed $n-1$ times, and there are 2 comparisons per iteration. Therefore (ignoring bookkeeping) there are $2n-2$ comparisons.

4. a) Since the list is in order, all the occurrences appear consecutively. Thus the output of our algorithm will be a pair of numbers, *first* and *last*, which give the first location and the last location of occurrences of x, respectively. All the numbers between *first* and *last* are also locations of appearances of x. If there are no appearances of x, we set *first* equal to 0 to indicate this fact.

```
procedure $all(x, a_1, a_2, \ldots, a_n$ : integers, with $a_1 \geq a_2 \geq \cdots \geq a_n)$
$i := 1$
while $i \leq n$ and $a_i < x$
        $i := i + 1$
if $i = n + 1$ then $first := 0$
else if $a_i > x$ then $first := 0$
else
begin
        $first := i$
        $i := i + 1$
        while $i \leq n$ and $a_i = x$
                $i := i + 1$
        $last := i - 1$
end
```

b) The number of comparisons depends on the data. Roughly speaking, in the worst case we have to go all the way through the list. This requires that x be compared with each of the elements, a total of n comparisons (not including bookkeeping). The situation is really a bit more complicated than this, but in any case the answer is $O(n)$.

6. a) We follow the instructions given. If n is odd then we start the loop at $i = 2$, and if n is even then we start the loop at $i = 3$. Within the loop, we compare the next two elements to see which is larger and which is smaller. The larger is possibly the new maximum, and the smaller is possibly the new minimum.

b)
```
procedure $clever\ smallest\ and\ largest(a_1, a_2, \ldots, a_n$ : integers)
if $n$ is odd then
begin
        $min := a_1$
        $max := a_1$
end
else if $a_1 < a_2$ then
```

```
begin
    $min := a_1$
    $max := a_2$
end
else
begin
    $min := a_2$
    $max := a_1$
end
if $n$ is odd then $i := 2$ else $i := 3$
while $i < n$
begin
    if $a_i < a_{i+1}$ then
    begin
        $smaller := a_i$
        $bigger := a_{i+1}$
    end
    else
    begin
        $smaller := a_{i+1}$
        $bigger := a_i$
    end
    if $smaller < min$ then $min := smaller$
    if $bigger > max$ then $max := bigger$
    $i := i + 2$
end { $min$ is the smallest integer among the input, and $max$ is the largest }
```

c) If n is even, then pairs of elements are compared (first with second, third with fourth, and so on), which accounts for $n/2$ comparisons, and there are an additional $2((n/2) - 1) = n - 2$ comparisons to determine whether to update *min* and *max*. This gives a total of $(3n - 4)/2$ comparisons. If n is odd, then there are $(n - 1)/2$ pairs to compare and $2((n - 1)/2) = n - 1$ comparisons for the updates, for a total of $(3n - 3)/2$. Note that in either case, this total is $\lceil 3n/2 \rceil - 2$ (see Exercise 7).

8. The naive approach would be to keep track of the largest element found so far and the second largest element found so far. Each new element is compared against the largest, and if it is smaller also compared against the second largest, and the "best-so-far" values are updated if necessary. This would require about $2n$ comparisons in all. We can do it more efficiently by taking Exercise 6 as a hint. If n is odd, set l to be the first element in the list, and set s to be $-\infty$. If n is even, set l to be the larger of the first two elements and s to be the smaller. At each stage, l will be the largest element seen so far, and s the second largest. Now consider the remaining elements two by two. Compare them and set a to be the larger and b the smaller. Compare a with l. If $a > l$, then a will be the new largest element seen so far, and the second largest element will be either l or b; compare them to find out which. If $a < l$, then l is still the largest element, and we can compare a and s to determine the second largest. Thus it takes only three comparisons for every pair of elements, rather than the four needed with the naive approach. The counting of comparisons is exactly the same as in Exercise 6: $\lceil 3n/2 \rceil - 2$.

10. We start with the solution to Exercise 37 in Section 3.1 and modify it to alternately examine the list from the front and from the back. The variables *front* and *back* will show what portion of the list still needs work. (After the k^{th} pass from front to back, we know that the final k elements are in their correct positions, and after the k^{th} pass from back to front, we know that the first k elements are in their correct positions.) The outer **if** statement takes care of changing directions each pass.

```
procedure shakersort($a_1, \ldots, a_n$)
front := 1
back := $n$
still_interchanging := true
while front < back and still_interchanging
        if $n$ + back + front is odd then
        begin {process from front to back}
                still_interchanging := false
                for $j$ := front to back − 1
                        if $a_j > a_{j+1}$ then
                        begin
                                still_interchanging := true
                                interchange $a_j$ and $a_{j+1}$
                        end
                back := back − 1
        end
        else {process from back to front}
        begin
                still_interchanging := false
                for $j$ := back down to front + 1
                        if $a_{j-1} > a_j$ then
                        begin
                                still_interchanging := true
                                interchange $a_{j-1}$ and $a_j$
                        end
                front := front + 1
        end {$a_1, \ldots, a_n$ is in nondecreasing order}
```

12. Lists that are already in close to the correct order will have few items out of place. One pass through the shaker sort will then have a good chance of moving these items to their correct positions. If we are lucky, significantly fewer than $n-1$ passes through the list will be needed.

14. Since $8x^3 + 12x + 100\log x \le 8x^3 + 12x^3 + 100x^3 = 120x^3$ for all $x > 1$, the conclusion follows by definition.

16. This is a sum of n things, each of which is no larger than $2n^2$. Therefore the sum is $O(2n^3)$, or more simply, $O(n^3)$. This is the "best" possible answer.

18. Let us look at the ratio $n^n/n!$. We can write this as

$$\frac{n}{n} \cdot \frac{n}{n-1} \cdot \frac{n}{n-2} \cdots \frac{n}{2} \cdot \frac{n}{1}.$$

Each factor is greater than or equal to 1, and the last factor is n. Therefore the ratio is greater than or equal to n. In particular, it cannot be bounded above by a constant C. Therefore the defining condition for n^n being $O(n!)$ cannot be met.

20. Let $q = \left\lceil \frac{a}{d} - \frac{1}{2} \right\rceil$ and $r = a - dq$. Then we have forced $a = dq + r$, so it remains to prove that $-d/2 < r \le d/2$. Now since $q - 1 < \frac{a}{d} - \frac{1}{2} \le q$, we have (by multiplying through by d and adding $d/2$) $dq - \frac{d}{2} < a \le dq + \frac{d}{2}$, so $-\frac{d}{2} < a - dq \le \frac{d}{2}$, as desired.

22. There is one 0 at the end of this number for every factor of 2 in all of the numbers from 1 to 100. We count them as follows. All the even numbers have a factor of 2, and there are $100/2 = 50$ of these. All the multiples of 4 have another factor of 2, and there are $100/4 = 25$ of these. All the multiples of 8 have another factor of 2, and there are $\lfloor 100/8 \rfloor = 12$ of these, and so on. Thus the answer is $50 + 25 + 12 + 6 + 3 + 1 = 97$.

24. We need to divide successively by 233, 144, 89, 55, 34, 21, 13, 8, 5, 3, 2, and 1, a total of 12 divisions.

26. **a)** The first statement is clear. For the second, if a and b are both even, then certainly 2 is a factor of their greatest common divisor, and the complementary factor must be the greatest common divisor of the numbers obtained by dividing out this 2. For the third statement, if a is even and b is odd, then the factor of 2 in a will not appear in the greatest common divisor, so we can ignore it. Finally, the last statement follows from Lemma 1 in Section 3.5, taking $q = 1$ (despite the notation, nothing in Lemma 1 required q to be the quotient).

b) All the steps involved in implementing part (a) as an algorithm require only comparisons, subtractions, and divisions of even numbers by 2. Since division by 2 is a shift of one bit to the right, only the operations mentioned here are used. (Note that the algorithm needs two more reductions: if a is odd and b is even, then $\gcd(a, b) = \gcd(a, b/2)$, and if $a < b$, then interchange a and b.)

c) We show the operation of the algorithm as a string of equalities; each equation is one step.

$$\begin{aligned}\gcd(1202, 4848) &= \gcd(4848, 1202) = 2\gcd(2424, 601) = 2\gcd(1212, 601) = 2\gcd(606, 601)\\ &= 2\gcd(303, 601) = 2\gcd(601, 303) = 2\gcd(298, 303) = 2\gcd(303, 298)\\ &= 2\gcd(303, 149) = 2\gcd(154, 149) = 2\gcd(77, 149) = 2\gcd(149, 77)\\ &= 2\gcd(72, 77) = 2\gcd(77, 72) = 2\gcd(77, 36) = 2\gcd(77, 18)\\ &= 2\gcd(77, 9) = 2\gcd(68, 9) = 2\gcd(34, 9) = 2\gcd(17, 9)\\ &= 2\gcd(8, 9) = 2\gcd(9, 8) = 2\gcd(9, 4) = 2\gcd(9, 2)\\ &= 2\gcd(9, 1) = 2\gcd(8, 1) = 2\gcd(4, 1) = 2\gcd(2, 1)\\ &= 2\gcd(1, 1) = 2\end{aligned}$$

28. We can give a nice proof by contraposition here, by showing that if n is not prime, then the sum of its divisors is not $n + 1$. There are two cases. If $n = 1$, then the sum of the divisors is $1 \neq 1 + 1$. Otherwise n is composite, so can be written as $n = ab$, where both a and b are divisors of n different from 1 and from n (although it might happen that $a = b$). Then n has at least the three distinct divisors 1, a, and n, and their sum is clearly not equal to $n + 1$. This completes the proof by contraposition. One should also observe that the converse of this statement is also true: if n is prime, then the sum of its divisors is $n + 1$ (since its only divisors are 1 and itself).

30. **a)** Each week consists of seven days. Therefore to find how many (whole) weeks there are in n days, we need to see how many 7's there are in n. That is exactly what n **div** 7 tells us.

b) Each day consists of 24 hours. Therefore to find how many (whole) days there are in n hours, we need to see how many 24's there are in n. That is exactly what n **div** 24 tells us.

32. We need to arrange that every pair of the four numbers has a factor in common. There are six such pairs, so let us use the first six prime numbers as the common factors. Call the numbers a, b, c, and d. We will give a and b a common factor of 2; a and c a common factor of 3; a and d a common factor of 5; b and c a common factor of 7; b and d a common factor of 11; and c and d a common factor of 13. The simplest way to accomplish this is to let $a = 2 \cdot 3 \cdot 5 = 30$; $b = 2 \cdot 7 \cdot 11 = 154$; $c = 3 \cdot 7 \cdot 13 = 273$; and $d = 5 \cdot 11 \cdot 13 = 715$. The numbers are mutually relatively prime, since no number is a factor of all of them (indeed, each prime is a factor of only two of them). Many other examples are possible, of course.

34. If $x \equiv 3 \pmod 9$, then $x = 3 + 9t$ for some integer t. In particular this equation tells us that $3 \mid x$. On the other hand the first congruence says that $x = 2 + 6s = 2 + 3 \cdot (2s)$ for some integer s, which implies that the remainder when x is divided by 3 is 2. Obviously these two conclusions are inconsistent, so there is no simultaneous solution to the two congruences.

36. **a)** There are two things to prove here. First suppose that $\gcd(m_1, m_2) \mid a_1 - a_2$; say $a_1 - a_2 = k \cdot \gcd(m_1, m_2)$. By Theorem 1 in Section 3.7 there are integers s and t such that $\gcd(m_1, m_2) = sm_1 + tm_2$. Multiplying both sides by k and substituting into our first equation we have $a_1 - a_2 = ksm_1 + ktm_2$, which can be rewritten as $a_1 - ksm_1 = a_2 + ktm_2$. This common value is clearly congruent to a_1 modulo m_1 and congruent to a_2 modulo m_2, so it is a solution to the given system. Conversely, suppose that there is a solution x to the system. Then $x = a_1 + sm_1 = a_2 + tm_2$ for some integers s and t. This says that $a_1 - a_2 = tm_2 - sm_1$. But $\gcd(m_1, m_2)$ divides both m_1 and m_2 and therefore divides the right-hand side of this last equation. Therefore it also divides the left-hand side, $a_1 - a_2$, as desired.
b) We follow the idea sketched in Exercises 23 and 24 of Section 3.7. First we show that if $a \equiv b \pmod{m_1}$ and $a \equiv b \pmod{m_2}$, then $a \equiv b \pmod{\text{lcm}(m_1, m_2)}$. The first hypothesis says that $m_1 \mid a - b$; the second says that $m_2 \mid a - b$. Therefore $a - b$ is a common multiple of m_1 and m_2. If $a - b$ were not also a multiple of $\text{lcm}(m_1, m_2)$, then $(a - b) \textbf{ mod } \text{lcm}(m_1, m_2)$ would be a common multiple as well, contradicting the definition of $\text{lcm}(m_1, m_2)$. Therefore $a - b$ is a multiple of $\text{lcm}(m_1, m_2)$, i.e., $a \equiv b \pmod{\text{lcm}(m_1, m_2)}$. Now suppose that there were two solutions to the given system of congruences. By what we have just proved, since these two solutions are congruent modulo m_1 (since they are both congruent to a_1) and congruent modulo m_2 (since they are both congruent to a_2), they must be congruent to each other modulo $\text{lcm}(m_1, m_2)$. That is precisely what we wanted to prove.

38. First note that since both a and b must be greater than 1, the sequences $\lfloor ka \rfloor$ and $\lfloor kb \rfloor$ do not list any positive integer twice. The issue is whether any positive integer is listed in both sequences, or whether some positive integer is omitted altogether. Let $N(x, n)$ denote the number of positive integers in the set $\{\, \lfloor kx \rfloor \mid k \text{ is a positive integer} \,\}$ that are less than or equal to n. Then it is enough to prove that $N(a, n) + N(b, n) = n$ for all positive integers n. (That way no positive integer could be left out or appear twice when we consider all the numbers $\lfloor ka \rfloor$ and $\lfloor kb \rfloor$.) Now $N(a, n)$ is the number of positive integers k for which $\lfloor ka \rfloor \le n$, which is just the number of positive integers k for which $ka < n + 1$, since a is irrational, and this is clearly $\lfloor (n+1)/a \rfloor$. We have a similar result for b. Let $f(x)$ denote the fractional part of x (i.e., $f(x) = x - \lfloor x \rfloor$). Then we have

$$N(a, n) + N(b, n) = \left\lfloor \frac{n+1}{a} \right\rfloor + \left\lfloor \frac{n+1}{b} \right\rfloor = \frac{n+1}{a} - f\left(\frac{n+1}{a}\right) + \frac{n+1}{b} - f\left(\frac{n+1}{b}\right).$$

But the sum of the first and third terms of the right-hand side here is $n + 1$, since we are given that $(1/a) + (1/b) = 1$. The second and fourth terms are each fractions strictly between 0 and 1, and the entire expression is an integer, so they must sum to 1. Therefore the displayed value is $n + 1 - 1 = n$, as desired.

40. If there were integer solutions to this equation, then by definition we would have $x^2 \equiv 2 \pmod 5$. However we easily compute (as in Exercise 24 in Section 3.4) that the square of an integer of the form $5k$ is congruent to 0 modulo 5; the square of an integer of the form $5k + 1$ is congruent to 1 modulo 5; the square of an integer of the form $5k + 2$ is congruent to 4 modulo 5; the square of an integer of the form $5k + 3$ is congruent to 4 modulo 5; and the square of an integer of the form $5k + 4$ is congruent to 1 modulo 5. This is a contradiction, so no solutions exist.

42. Since $\mathbf{A}$ is the matrix defined by $a_{ii} = c$ and $a_{ij} = 0$ for $i \ne j$, it is easy to see from the definition of multiplication that $\mathbf{AB}$ and $\mathbf{BA}$ are both the same as $\mathbf{B}$ except that every entry has been multiplied by c. Therefore these two matrices are equal.

44. We just use Algorithm 1 in Section 3.8, where $\mathbf{A}$ and $\mathbf{B}$ are now $n \times n$ upper triangular matrices, by replacing m by n in line 1, and having q iterate only from i to j, rather than from 1 to k. (Actually even more efficiency is possible if we store only the nonzero portion of the matrices—see solution to Exercise 45.)

46. See the solution to Exercise 45. Looking at the nested loops, we see that the number of multiplications is given by the following expression: $[1+2+\cdots+n]+[1+2+\cdots(n-1)]+\cdots+[1]$. To simplify this, we need some results from Section 4.1, namely Example 1, which says that the sum of the first k positive integers is $k(k+1)/2$, and Exercise 15, which says that the sum $1\cdot 2+2\cdot 3+3\cdot 4+\cdots+k(k+1)=k(k+1)(k+1)/3$. Doing the algebra, we obtain $n(n+1)(n+2)/6$ as the total number of multiplications.

48. There are five ways to parenthesize the calculation $\mathbf{ABCD}$, namely as $(\mathbf{AB})(\mathbf{CD})$, as $((\mathbf{AB})\mathbf{C})\mathbf{D}$, as $(\mathbf{A}(\mathbf{BC}))\mathbf{D}$, as $\mathbf{A}((\mathbf{BC})\mathbf{D})$, and as $\mathbf{A}(\mathbf{B}(\mathbf{CD}))$. We can compute the number of multiplications for each and compare. These numbers are (in thousands) 108, 117, 80, 44, and 81. Thus the most efficient method is the fourth, using 44,000 multiplications.

50. The greedy algorithm in this case will produce the base c expansion for the number of cents required (except that for amounts greater than or equal to c^{k+1}, the c^k coins must be used rather than nonexistent c^i coins for $i>k$). Since such expansions are unique if each digit (other than the digit in the c^k place) is less than c, the only other ways to make change would involve using c or more coins of a given denomination, and this would obviously not be minimal, since c coins of denomination c^i could be replaced by one coin of denomination c^{i+1}.

52. As we see from Exercise 51, at most n questions (guesses) are needed. Furthermore, at least this many yes/no questions are needed as well, since if we asked fewer questions, then by the pigeonhole principle, two numbers would produce the same set of answers and we would be unable to guess the number accurately. Thus the complexity is n questions. (The case $n=0$ is not included, since in that case no questions are needed.) We are assuming throughout this exercise and the previous one that the inclusive sense of "between" was intended.