CHAPTER 3

Section 3.1

1. \( \text{max} = 1, i = 2, \text{max} := 0, i := 3, \text{max} := 12, i := 4, i := 5, i := 6, j := 7, \text{max} := 14, i := 8, j := 9, i := 10, \text{max} := 11, i := 11 \)

3. procedure \( p(a_1, \ldots, a_n: \text{integers}) \)
   
   \[
   \text{sum} := a_1
   \text{for } i := 2 \text{ to } n
   \text{sum} := \text{sum} + a_i
   \text{return sum}
   \]

5. procedure duplicates\( (a_1, a_2, \ldots, a_n: \text{integers in nondecreasing order}) \)
   
   \[
   k := 0 \quad (\text{this counts the duplicates})
   j := 2
   \text{while } j \leq n
   \text{if } a_j = a_{j-1}
   k := k + 1
   \text{while } j \leq n \text{ and } a_j = c_k
   j := j + 1
   \]
   \( k \) is the desired list

7. procedure last even location\( (a_1, a_2, \ldots, a_n: \text{integers}) \)
   
   \[
   k := 0
   \text{for } i := 1 \text{ to } n
   \text{if } a_i \text{ is even then } i := i
   \text{return } k \quad (k = 0 \text{ if there are no evens})
   \]

9. procedure palindrome check\( (a_1, a_2, \ldots, a_n: \text{string}) \)
   
   \[
   \text{answer} := \text{true}
   \text{for } i := 1 \text{ to } (n/2)
   \text{if } a_i \neq a_{n-i+1} \text{ then } \text{answer} := \text{false}
   \text{return } \text{answer}
   \]

11. procedure interchange\( (x, y: \text{real numbers}) \)
    
    \[
    \text{i} := x
    x := y
    y := i
    \]

The minimum number of assignments needed is three.

13. \( i := 1, j := 2, i := 3, j := 4, i := 5, i := 6, j := 7, \text{location} := 7; \) binary search; \( i := 1, j := 8, i := 4, j := 5, m := 6, i := 7, j := 7, \text{location} := 7 \)

15. \( i := 1, j := 1, \text{while } x > a_j \text{ then } j := j + 1 \)
    
    \[
    a_{i+1} := x + 1
    i := 1
    \text{while } x > a_i
    i := i + 1
    \text{for } j := 1 \text{ to } a_i
t := a_j
    a_{j+1} := t
    a_i := x
    (x has been inserted into correct position)\]

17. procedure first largest\( (a_1, \ldots, a_n: \text{integers}) \)
    
    \[
    \text{max} := a_1
    \text{location} := 1
    \text{for } i := 2 \text{ to } n
    \text{if } a_i < \text{max} \text{ then}
    \text{max} := a_i
    \text{location} := i
    \text{return location}
    \]

19. procedure mean-median-max-min\( (a, b, c: \text{integers}) \)
    
    \[
    \text{mean} := (a + b + c)/3
    \text{(the six different orderings of } a, b, c \text{ with respect to } \geq \text{ will be handled separately)}
    \text{if } x \geq y \text{ then}
    \text{if } y \geq z \text{ then}
    \text{median} := a; \text{min} := c
    \]
    \( \text{(The rest of the algorithm is similar.)} \)

21. procedure first three\( (a_1, a_2, \ldots, a_n: \text{integers}) \)
    
    \[
    \text{if } a_1 > a_2 \text{ then interchange } a_1 \text{ and } a_2
    \text{if } a_2 > a_3 \text{ then interchange } a_2 \text{ and } a_3
    \text{if } a_1 > a_3 \text{ then interchange } a_1 \text{ and } a_3
    \]

23. procedure \( f \text{: function from } A \text{ to } B \) where \( A = (a_1, \ldots, a_n), B = (b_1, \ldots, b_m), a_1, \ldots, a_n, b_1, \ldots, b_m \) are integers)
    
    \[
    \text{for } i := 1 \text{ to } n
    \text{hit}(a_i) := 0
    \text{count} := 0
    \text{for } j := 1 \text{ to } m
    \text{if } \text{hit}(a_i) = 0 \text{ then}
    \text{hit}(a_i) := 1
    \text{count} := \text{count} + 1
    \text{if count} = m \text{ then return true else return false}
    \]

25. procedure ones\( (x: \text{bit string}, a = a_2, \ldots, a_1) \)
    
    \[
    \text{count} := 0
    \text{for } i := 1 \text{ to } n
    \text{if } a_i = 1 \text{ then}
    \text{count} := \text{count} + 1
    \text{return count}
    \]

27. procedure ternary search\( (x: \text{integer}, a_1, a_2, \ldots, a_n: \text{increasing integers}) \)
    
    \[
    i := 1
    j := n
    \text{while } i < j \text{ do }
    l := ((i + j)/3)
    u := ((2i + j)/3)
    \text{if } x > a_u \text{ then } i := u + 1
    \text{else if } x < a_l \text{ then } j := l - 1
    \text{else if } x = a_m \text{ then location} := i
    \text{else if } x = a_u \text{ then location} := j
    \text{else if } x = a_l \text{ then location} := j
    \]
33. procedure find a mode(a1, a2, ..., an; nondecreasing integers)
    modecount := 0
    i := 1
    while i ≤ n
        value := ai
        count := count + 1
        i := i + 1
    if count = modecount then
        modecount := count
        mode := value
    return mode

31. procedure find duplicate(a1, a2, ..., an; positive integers)
    location := 0
    i := 2
    while i ≤ n and location = 0
        if ai < ai+1 then location := i
        else i := i + 1
    return location

33. procedure find decrease(a1, a2, ..., an; positive integers)
    location := 0
    i := 2
    while i ≤ n and location = 0
        if ai < ai-1 then location := i
        else i := i + 1
    return location

35. At the end of the first pass, 1, 3, 5, 4, 7; at the end of the second pass, 1, 3, 4, 5, 7; at the end of the third pass, 1, 3, 4, 5, 7; at the end of the fourth pass, 1, 3, 4, 5, 7.

39. At the end of the first, second, and third passes: 1, 3, 5, 7, 4; at the end of the fourth pass: 1, 3, 4, 5, 7, 2, 1, 2, 4, 3, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5.

41. a) 1, 5, 4, 3, 2, 1, 2, 4, 3, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5
    b) 1, 4, 3, 2, 5, 1, 2, 4, 3, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5
    c) 3, 1, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5
    d) 2

43. We carry out the linear search algorithm given as Algorithm 2 in this section, except that we replace s ≠ a0 by s < a0, and we replace the else clause with else location := n + 1.

45. 2 + 3 + 4 + ... + n = (n2 + n) / 2

47. Find the location for the 2 in the list 3 (one comparison), and insert it in front of the 3, so the list now reads 2, 3, 4, 5, 1, 6. Find the location for the 4 (compare it to the 2 and then the 3), and insert it, leaving 2, 3, 4, 5, 1, 6. Find the location for the 5 (compare it to the 3 and then the 4), and insert it, leaving 2, 3, 4, 5, 1, 6. Find the location for the 6 (compare it to the 3 and then the 4 and then the 5), and insert it, giving the final answer 1, 2, 3, 4, 5, 6.
set woman’s proposal list to be empty while rejected men remain
for $i = 1$ to $n$
If man $i$ is marked rejected then add $i$ to the proposal list for the woman $j$ who ranks highest on his preference list but does not appear on his rejection list, and mark $j$ as not rejected.
If woman $j$’s proposal list is nonempty then remove from $j$’s proposal list all men $i$ except the man $k$ who ranks highest on her preference list, and for each such man $k$ mark him as rejected and add $j$ to his rejection list
for $j = 1$ to $m$
yield the match with the one man on $j$’s proposal list.
This matching is stable.
If the assignment is not stable, then there is a man $w$ and a woman $v$ such that $w$ prefers $v$ to the woman $w’$ with whom he is matched, and $w’$ prefers $w$ to the man with whom she is matched. But $w’$ must have proposed to $w’$ before he proposed to $w$, because he prefers the former. Because $w’$ did not end up matched with $w$, she must have rejected him. Women re- ject a suitor only when they get a better proposal, and they eventually get matched with a pending suitor, so the woman with whom $w$ is matched must be better in her eyes than $w’$, contradicting our original assumption. Therefore the marriage is stable.
Run the two programs on their inputs concur- rently and report which one halts.

Section 3.2
1. The choices of $C$ and $k$ are not unique. $a) C = 1, k = 10$ $b) C = 4, k = \lceil \log n \rceil$ $c) C = 5, k = 1$ $d) C = 1, k = 0$ $f) C = 1, k = 2$ $g) C = 7$ $h) C = 2, k = 0$ $i) C = 5, k = 6$ $j) C = 1, k = 1$ $k) C = 3, k = 1, k’ = 1$ $l) C = 1, n$ $m) C = 2, k = 1$ $n) C = 3, k = 1, n = 1$ $o) C = 2, k = 1$ $p) C = 1, k = 1$ $q) C = 3, k = 1, n = 1$ $r) C = 1, k = 1$ $s) C = 3, k = 1$ $t) C = 3, k = 1$ $u) C = 3, k = 1$ $v) C = 3, k = 1$ $w) C = 3, k = 1$ $x) C = 3, k = 1$ $y) C = 3, k = 1$ $z) C = 3, k = 1$ $A) k = 1$ $B) k = 1$ $C) k = 1$ $D) k = 1$ $E) k = 1$ $F) k = 1$ $G) k = 1$ $H) k = 1$ $I) k = 1$ $J) k = 1$ $K) k = 1$ $L) k = 1$ $M) k = 1$ $N) k = 1$ $O) k = 1$ $P) k = 1$ $Q) k = 1$ $R) k = 1$ $S) k = 1$ $T) k = 1$ $U) k = 1$ $V) k = 1$ $W) k = 1$ $X) k = 1$ $Y) k = 1$ $Z) k = 1$
2. The choices of $C$ and $k$ are not unique. $a) C = 1, k = 1$ $b) C = 1, k = 2$ $c) C = 1, k = 3$ $d) C = 1, k = 4$ $e) C = 1, k = 5$ $f) C = 1, k = 6$ $g) C = 1, k = 7$ $h) C = 1, k = 8$ $i) C = 1, k = 9$ $j) C = 1, k = 10$ $k) C = 1, k = 11$ $l) C = 1, k = 12$ $m) C = 1, k = 13$ $n) C = 1, k = 14$ $o) C = 1, k = 15$ $p) C = 1, k = 16$ $q) C = 1, k = 17$ $r) C = 1, k = 18$ $s) C = 1, k = 19$ $t) C = 1, k = 20$ $u) C = 1, k = 21$ $v) C = 1, k = 22$ $w) C = 1, k = 23$ $x) C = 1, k = 24$ $y) C = 1, k = 25$ $z) C = 1, k = 26$
3. By definition there are positive constraints $f(\theta)x|\leq g(x)$ and $|f(\theta)x|\leq h(x)$ such that $f(\theta)x|\leq g(x)$ for $x > \theta$. Thus, $f(\theta)x|\leq g(x)$ and $g(x)$ is $O(\theta f(x))$. Con- versely, suppose that $f(\theta)x|\leq g(x)$ and $g(x)$ is $O(\theta f(x))$. Then there are constants $C_1, C_2, C_3, C_4$, and $k_1$ and $k_2$ such that $|f(\theta)x|\leq C_1|g(x)|$ for all $x > k_1$ and $|f(\theta)x|\leq C_2|g(x)|$ for all $x > k_2$. It follows that $|f(\theta)x|\leq (C_1C_2)|g(x)|$ for all $x > k$ whenever $k = \max(k_1, k_2)$. Hence $f(\theta)x|\leq (C_1C_2)|g(x)|$ for $x > k$. Thus, $f(\theta)x|\leq g(x)$ and $g(x)$ is $O(\theta f(x))$. Con- versely, suppose that $f(\theta)x|\leq g(x)$ and $g(x)$ is $O(\theta f(x))$. Then there are constants $C_1, C_2, C_3, C_4$, and $k_1$ and $k_2$ such that $|f(\theta)x|\leq C_1|g(x)|$ for all $x > k_1$ and $|f(\theta)x|\leq C_2|g(x)|$ for all $x > k_2$. It follows that $|f(\theta)x|\leq (C_1C_2)|g(x)|$ for all $x > k$ whenever $k = \max(k_1, k_2)$. Conversely, if there are positive constants $C_1, C_2, C_3$, and $C_4$ such that $C_1|g(x)|\leq |f(\theta)x|\leq C_2|g(x)|$ for $x > k$, then taking $k_1 = k_2 = k$ shows that $f(\theta)x|\leq (C_1C_2)|g(x)|$.