Chapter 2:

**Exercise 1a.** Set the value of area to \( \frac{1}{2} b \cdot h \)

**Exercise 1c.** Set the value of \( \text{FlyingTime} \) to \( M/AveSpeed \)

**Exercise 4.** Algorithm:

1. Get the values of \( L, W \) and \( C \)
2. Set the value of \( \text{Area} \) to \( L \cdot W/9 \) to get the area in square yards
3. Set the value of \( \text{TotalCost} \) to \( 1.06 \cdot \text{Area} \cdot C \)
4. Print the value of \( \text{TotalCost} \)

**Exercise 9.** Algorithm:

1. Repeat Steps 2 - 5 until \( \text{TotalCost} < 1000 \)
2. Get a value for \( L, W \) and \( C \)
3. Set the value of \( \text{Area} \) to \( L \cdot W/9 \)
4. Set the value of \( \text{Total Cost} \) to \( 1.06 \cdot \text{Area} \cdot C \)
5. Print the value of \( \text{TotalCost} \)

**Exercise 11.** Algorithm:

1. Repeat Steps 2 - 17 until \( \text{Response} = \text{‘No’} \)
2. Get values of \( \text{Hours} \) and \( \text{Rate} \)
3. If \( \text{Hours} > 54 \) then
4. \( \text{DoubleTime} = \text{Hours} - 54 \)
5. \( \text{TimeAndHalf} = 14 \)
6. \( \text{Regular} = 40 \)
7. else if \( \text{Hours} > 40 \) then
8. \( \text{DoubleTime} = 0 \)
9. \( \text{TimeAndHalf} = \text{Hours} - 40 \)
10. \( \text{Regular} = 40 \)
11. else \( \text{DoubleTime} = 0 \)
12. \( \text{TimeAndHalf} = 0 \)
13. \( \text{Regular} = \text{Hours} \)
14. \( \text{GrossPay} = \text{Rate} \cdot \text{Hours} + 1.5 \cdot \text{Rate} \cdot \text{TimeAndHalf} + 2.0 \cdot \text{Rate} \cdot \text{DoubleTime} \)
15. Print the value of \( \text{GrossPay} \)
16. Print the message “Do you wish to do another computation?”
17. Get the value of \( \text{Response} \)
Exercise 14. Assume that *FindLargest* is now a primitive operation in our pseudocode and use it to repeatedly remove the largest element from the list until we reach the median.

Algorithm:

- Step 1. Get the value of *N*, and the values *L*<sub>1</sub>, *L*<sub>2</sub>, ..., *L*<sub>*N* in the list
- Step 2. If *N* is even, then set *M* = *N*/2
- Step 3. else let *M* = *(N + 1)/2
- Step 4. Repeat Steps 5 to 10 until *N* < *M*
- Step 5. Use *FindLargest* to find the location *Loc* of the largest element in the list *L*<sub>1</sub>, *L*<sub>2</sub>, ..., *L*<sub>*N*  
- Step 6. Exchange *L*<sub>*Loc*</sub> and *L*<sub>*N*</sub> as follows
- Step 7. *Temp* = *L*<sub>*N*</sub>
- Step 8. *L*<sub>*N*</sub> = *L*<sub>*Loc*</sub>
- Step 9. *L*<sub>*Loc*</sub> = *Temp*
- Step 10. Set *N* to *N* − 1 and shorten the list
- Step 11. Print the message “The median is:”
- Step 12. Print the value of *L*<sub>*M*</sub>
- Step 13. Stop

Chapter 3.

Exercise 4. *Legit* = 6

```
1 2 3 4 5 6 6 6 6
```

Number of Copies: 16

Exercise 5. *Legit* = 6

```
1 6 2 5 3 4 5 0 6
```

Number of Copies: 3

Exercise 6. Once item *N* has been copied one cell left, it need not be copied again. Similarly, once item *N* − 1 has been copied one cell left, it need not be copied again. The value of *Legit* shows how many cells from the right have been copied.

Step 11 of the algorithm can be changed to

“Repeat Steps 12 and 13 until *Right* > *Legit* + 1”

Only 13 copies are done
**Exercise 14b.** The list after each exchange in *BubbleSort* is shown below.

```
 12  3  6  8  2  5  7  
  3  12  6  8  2  5  7  
  3  6  12  8  2  5  7  
  3  6  8  12  2  5  7  
  3  6  8  2  12  5  7  
  3  6  8  2  5  12  7  
  3  6  8  2  5  7  12  
  3  6  2  8  5  7  12  
  3  6  2  5  8  7  12  
  3  6  2  5  7  8  12  
  3  2  6  5  7  8  12  
  3  2  5  6  7  8  12  
  2  3  5  6  7  8  12  
```

Bubblesort required more exchanges than selection sort on the input above.

**Exercise 17c.** The names compared with *Emile* are

```
  John    Elsa    JoAnn
```

and there are only 3 comparisons made to determine that *Emile* is not in the list.

**Exercise 21.** The binary search tree is shown below.

```
  John
    /\  /\ 
  Elsa  Lee
    /\  /\ 
Arturo  JoAnn  Jose  Snyder
    /\  /\  /\  /\ 
    Tracy
```

Worst Case: 4 comparisons

Would occur if searching for Tracy