Solutions to End-of-Chapter Exercises

Chapter 1: An Introduction to Computer Science

1. There is no one correct answer. Common examples are the instructions for using a voice mail system, the instructions for opening a mail box lock, and the instructions for doing laundry.

2. A *heuristic* is a method for finding a reasonably close, "good enough" solution to a problem. It can be viewed as a rule-of-thumb, a method of approximation, an informal technique, or even a way to make an "educated guess." It differs from the concept of an algorithm in that it does not guarantee to produce an optimal solution, just to make a good faith attempt to locate a reasonable one. Heuristics are often used when executing an algorithm might be too time-consuming, and we only need an approximation to the correct answer.

An example of a heuristic for adding two 3-digit numbers, such as 234 + 567, might be:

1. Set the one and tens digit of both operands to 0
2. Increase the hundreds digit of the second operand by 1. These two steps result in changing the problem to the simpler one 200 + 600.
3. Add the hundreds digits, resulting in a final "answer" of 800.

Now, of course, this is not the correct answer, which is 801. But the result we get may be close enough for our needs, and it is certainly a lot easier to add a single column of numbers rather than three columns of numbers.

3. One may argue that the instruction is not well-ordered, since it is unclear whether one should enter the channel first or press CHAN first. Also, it may not be effectively computable if you desire to enter a channel that is out of the DVR's range.

4. (a) Sequential
(b) Conditional
(c) Sequential
(d) Iterative

5. Step 1: $carry = 0$, c_3 = ??, c_2 = ??, c_1 = ??, and c_0 = ??

Step 2: $i = 0$, all others unchanged

Step 4: $c_0 = 18$, all others unchanged

Step 5: $c_0 = 8$ and $carry = 1$, all others unchanged

Step 6: $i = 1$, $carry = 1$, c_3 = ??, c_2 = ??, c_1 = ??, and $c_0 = 8$

Step 4: $c_1 = 7$, all others unchanged

Step 5: $carry = 0$, all others unchanged

Step 6: $i = 2$, $carry = 0$, c_3 = ??, c_2 = ??, $c_1 = 7$, and $c_0 = 8$

Step 4: $c_1 = 1$, all others unchanged

Step 5: *carry* = 0, all others unchanged

Step 6: $i = 3$, $carry = 0$, $c_3 = ??$, $c_2 = 1$, $c_1 = 7$, and $c_0 = 8$

Step 7: $c_3 = 0$, $c_2 = 1$, $c_1 = 7$, and $c_0 = 8$

Step 8: Print out 0178.

6. Replace Step 8 with the following steps:

Step 8: Set the value of *i* to *m*

Step 9: Repeat step 10 until either c_i is not equal to 0 or $i < 0$

Step 10: Subtract 1 from *i*, moving one digit to the right

Step 11: If $i \geq 0$ then print $c_i c_{i-1} \ldots c_0$

7. Assume that *a* has *n* digits $a_{n-1}, \ldots, a_0$, and *b* has *m* digits, $b_{m-1}, \ldots, b_0$, with *n* not necessarily equal to *m*. Add an operation at the beginning of the algorithm that resets the two numbers to the same number of digits by adding non-significant leading zeros to the shorter one. We can then reuse the algorithm of Figure 1.2.

```
If (m > n)  then
    Set i to 0
    While (n+i < m)
        Add a leading zero to the number at position  $a_{n+i}$
        Increment i by 1
    End of the loop
Else
    If (n > m)
        Set i to 0
        While (m+i < n)
            Add a leading zero to the number at position  $b_{m+i}$
            Increment i by 1
        End of the loop
```

We have now made the two numbers equal in length. All we need do now is set the variable *m* to the larger of the two values:

Set m to the larger of m and n.

The addition algorithm in Figure 1.2 will now work correctly. Note that if m and n are equal in value, neither of the Boolean expressions will be true, and neither of the conditional statements will be executed.

8. It is not effectively computable if $b^2 - 4ac < 0$ (since we cannot take the square root of a negative number if we are limited to real numbers) or if $a = 0$ (since we cannot divide by 0).

9. The first algorithm (Figure 1.3(a)) is a better general purpose algorithm. If you want to shampoo your hair any number *n* times you can change the 2 to *n*. You could even ask the shampooer to input the desired number *n* of washings. For the second algorithm you would have to rewrite the algorithm to repeat steps 4 and 5 998 more times.

10. (a) Trace:

```
Step 1:  $I = 32$, $J = 20$, and $R = ??$
Step 2:  $I = 32$, $J = 20$, and $R = 12$
Step 3:  $I = 20$, $J = 12$, and $R = 12$
Step 2:  $I = 20$, $J = 12$, and $R = 8$
Step 3:  $I = 12$, $J = 8$, and $R = 8$
Step 2:  $I = 12$, $J = 8$, and $R = 4$
Step 3:  $I = 8$, $J = 4$, and $R = 4$
Step 2:  $I = 8$, $J = 4$, and $R = 0$
Step 4:  Print $J = 4$
```

(b) At Step 2 we are asked to divide $I = 32$ by $J = 0$, which cannot be done. We can fix the problem by adding a step between Step 1 and Step 2 that says: If $J = 0$, then print "ERROR: division by 0" and Stop.

11. There are 25! possible paths to be considered. That is approximately 1.5×10^{25} different paths. The computer can analyze 10,000,000, or 10^7, paths per second. The number of seconds required to check all possible paths is about $1.5 \times 10^{25}/10^7$, or about 1.5×10^{18} seconds. That's roughly 10^{12} years: about a trillion years. This would not be a feasible algorithm.

12. A Multiplication Algorithm.

```
Given:  Two positive numbers $a$ and $b$
Wanted:  A number $c$ which contains the result of multiplying $a$ and $b$
Step 1:  Set the value of $c$ equal to 0
Step 2:  Set the value of $i$ equal to $b$
Step 3:  Repeat steps 4 and 5 until the value of $i$ is 0
Step 4:     Set the value of $c$ to be $c + a$

Step 5:     Subtract 1 from $i$

Step 6:  Print out the final answer $c$

Step 7:  Stop
```

This algorithm assumes that we know how to add two multiple-digit numbers together. We may assume this because we have the algorithm from the book which does exactly that.

13. The algorithm will work correctly only if all three numbers are unique. If two or more numbers are identical, none of the Boolean expressions will be true and nothing will be output. To make this a correct solution you either have to specify in the problem statement that the three numbers provided must all be distinct or (better) change all of the comparison operations to $\geq$ in place of $>$.

14. This is an essay question. Students may find excellent resources on the Internet.

15. If this problem is assigned, be sure to coordinate with your computing staff ahead of time for students to get the required information.

16. This is an essay question. Because this is a "hot" topic, a great deal of hype and hyperbole is available, as well as useful information. It might be a good opportunity to teach students about finding *reliable* sources on the Internet, and evaluating online and print source materials.

17. Like question 16 this is an essay question. Students may be familiar with Apple iCloud services for iPhone and iPad devices, so it might be a good opportunity to relate their answers to the services provided by Apple.

18. About 130 feet ((((700,000,000 chars/5 chars per word)/300 words per page)/300 pages per inch)/12 inch per foot)

Discussion of Challenge Work

1. We may perform subtraction, like addition, by subtracting one column at a time, starting with the rightmost column and working to the left. Since we know that the first number is larger than the second one, we know that we can always borrow from columns to the left of the current one. Therefore, if the upper number in the column (a_i) is smaller than the lower, we automatically borrow from the next column. We can do this by subtracting one from the a_{i+1} value of the column to the left. If the a_{i+1} value were already zero, then it would become -1. This automatically causes a borrow to occur on the next step. Here is the algorithm:

```
Step 1:  Set the value of $i$ equal to the value of 0

Step 2:  Repeat steps 3 to 6 until the value of $i$ is $m$
Step 3:     If $b_i \le a_i$ then
Step 4:         Set $c_i$ equal to $a_i - b_i$
            Otherwise ($b_i > a_i$)
Step 5:     Set $c_i$ equal to $(a_i + 10) – b_i$ and replace $a_{i+1}$ with $a_{i+1} – 1$
                (This amounts to a borrow of 1 from $a_{i+1}$ which adds 10 to $a_i$)
Step 6:     Add 1 to $i$ (moving us one column to the left)
Step 7:  Print out the final answer $c_{m-1}c_{m-2} . . . c_0$
Step 8:  Stop.
```

2. Students may need assistance finding or understanding other definitions from other sources.

Chapter 2: Algorithm Discovery and Design

1. (a) Set the value of *area* to $\frac{1}{2}(b \times h)$
(b) Set the value of *interest* to $I \times B$
Set the value of *FinalBalance* to $(1 + I) \times B$
(c) Set the value of *FlyingTime* to $M/AvgSpeed$

2. Algorithm:

```
Step 1:  Get values for $B, I,$ and $S$

Step 2:  Set the value of FinalBalance to $(1 + I/12)^{12}B$

Step 3:  Set the value of Interest to $FinalBalance - B$

Step 4:  Set the value of FinalBalance to $FinalBalance - S$

Step 5:  Print the message 'Interest Earned: '

Step 6:  Print the value of Interest

Step 7:  Print the message 'Final Balance: '

Step 8:  Print the value of FinalBalance
```

3. Algorithm:

```
Step 1:  Get values for $E1, E2, E3$ and $F$

Step 2:  Set the value of Ave to $(E1 + E2 + E3 + 2F)/5$

Step 3:  Print the value of Ave
```

4. Algorithm:

```
Step 1:  Get values for $P$ and $Q$

Step 2:  Set the value of Subtotal to $P \times Q$

Step 3:  Set the value of TotalCost to $(1.06) \times Subtotal$

Step 4:  Print the value of TotalCost
```

5. (a)

```
If $y \neq 0$ then
    Print the value of $(x/y)$

Else
    Print the message 'Unable to perform the division.'
```

(b)

```
If $r \geq 1.0$, then
    Set the value of Area to $\pi \times r^2$
    Set the value of $Circum$ to $2 \times \pi \times r$

Else
    Set the value of $Circum$ to $2 \times \pi \times r$
```

6. Algorithm:

```
Step 1:  Get a value for $B, I,$ and $S$

Step 2:  Set the value of $FinalBalance$ to $(1 + I/12)^{12}B$

Step 3:  Set the value of $Interest$ to $FinalBalance - B$

Step 4:  If $B < 1000$ then Set the value of $FinalBalance$ to $FinalBalance - S$

Step 5:  Print the message 'Interest Earned: '

Step 6:  Print the value of $Interest$

Step 7:  Print the message 'Final Balance: '

Step 8:  Print the value of $FinalBalance$
```

7. Algorithm:

```
Step 1:  Set the value of $i$ to 1

Step 2:  Set the values of $Won$, $Lost$, and $Tied$ all to 0

Step 3:  While $i \leq 10$ do

Step 4:      Get the value of $CSU_i$ and $OPP_i$

Step 5:      If $CSU_i > OPP_i$ then

Step 6:          Set the value of $Won$ to $Won + 1$

Step 7:      Else if $CSU_i < OPP_i$ then

Step 8:          Set the value of $Lost$ to $Lost + 1$
Step 9:     Else
Step 10:        Set the value of $Tied$ to $Tied + 1$
Step 11:    Set the value of $i$ to $i + 1$
         End of the While loop
Step 12:  Print the values of $Won$, $Lost$, and $Tied$
Step 13:  If $Won = 10$, then
Step 14:        Print the message, 'Congratulations on your undefeated season.'
```

8. Algorithm:

```
Step 1:  Set the value of $i$ to 1
Step 2:  Set the value of $Total$ to 0
Step 3:  While $i \leq 14$ do
Step 4:        Get the value of $E_i$
Step 5:        Set the value of $Total$ to $Total + E_i$
Step 6:        Set the value of $i$ to $i + 1$
         End of While loop
Step 7:  Get the value of $F$
Step 8:  Set the value of $Total$ to $Total + 2F$
Step 9:  Set the value of $Ave$ to $Total / 16$
Step 10: Print the value of $Ave$
```

9. Algorithm:

```
Step 1:  Set the value of $TotalCost$ to 0
Step 2:  Do
Step 3:        Get values for $P$ and $Q$
Step 4:        Set the value of $Subtotal$ to $P \times Q$
Step 5:        Set the value of $TotalCost$ to $TotalCost + (1.06) \times Subtotal$
```

```
          While ($TotalCost$ < 1000)
    Step 6:   Print the value of $TotalCost$
```

10. The tricky part is in steps 6 through 9. If you use no more than 1000 kilowatt hours in the month then you get charged $.06 for each. If you use more than 1000, then you get charged $.06 for the first 1000 hours and $.08 for each of the remaining hours. There are M_i – 1000 remaining hours, since M_i is the number of hours in the *i*th month. Also, remember that $KWBegin_i$ and $KWEnd_i$ are meter readings, so we can determine the total kilowatt-hours used for the whole year by subtracting the first meter reading ($KWBegin_1$) from the last ($KWEnd_{12}$).

```
    Step 1:  Set the value of $i$ to 1
    Step 2:  Set the value of $AnnualCharge$ to 0
    Step 3:  While $i \le 12$ do
    Step 4:      Get the value of $KWBegin_i$ and $KWEnd_i$
    Step 5:      Set the value of $M_i$ to $KWEnd_i - KWBegin_i$
    Step 6:      If $M_i \le 1000$ then
    Step 7:          Set $AnnualCharge$ to $AnnualCharge + (.06M_i)$
    Step 8:      Else
    Step 9:          Set $AnnualCharge$ to $AnnualCharge + (.06)1000$
                                              $+ (.08)(M_i - 1000)$
    Step 10:     Set the value of $i$ to $i + 1$
             End of While loop
    Step 11: Print the value of $AnnualCharge$
    Step 12: If ($KWEnd_{12} - KWBegin_1$) < 500, then
    Step 13:     Print the message 'Thank you for conserving electricity.'
```

11. Algorithm:

```
    Step 1:  Do
    Step 2:      Get the values of $HoursWorked$ and $PayRate$
    Step 3:      If $HoursWorked$ > 54 then
```

```
Step 4:             $DT = HoursWorked - 54$
Step 5:             $TH = 14$
Step 6:             $Reg = 40$
Step 7:         Else if $HoursWorked > 40$ then
Step 8:             $DT = 0$
Step 9:             $TH = HoursWorked - 40$
Step 10:            $Reg = 40$
Step 11:        Else ($HoursWorked \leq 40$)
Step 12:            $DT = 0$
Step 13:            $TH = 0$
Step 14:            $Reg = HoursWorked$
Step 15:        $GrossPay = PayRate \times Reg$
                    $+ 1.5 \times PayRate \times TH + 2 \times PayRate \times DT$
Step 16:        Print the value of GrossPay
Step 17:        Print the message 'Do you wish to do another computation?'
Step 18:        Get the value of Again
            While ($Again$ = yes)
```

12. Steps 1, 2, 5, 6, 7, and 9 are sequential operations and steps 4 and 8 are conditional operations. After their completion, the algorithm moves on to the step below it, so none of these could cause an infinite loop. Step 3, however, is a while loop, and it could possibly cause an infinite loop. The true/false looping condition is "*Found* = NO and $i \leq 10{,}000$." If NUMBER is ever found in the loop then *Found* gets set to YES, the loop stops, and the algorithm ends after executing steps 8 and 9. If NUMBER is never found, then 1 is added to *i* at each iteration of the loop. Since step 2 initializes *i* to 1, *i* will become 10,001 after the 10,000th iteration of the loop. At this point the loop will halt, steps 8 and 9 will be executed, and the algorithm will end.

13. Algorithm:

```
Step 1:  Get values for NUMBER, $T_1, \ldots T_{10000}$, and $N_1, \ldots, N_{10000}$
Step 2:  Set the value of i to 1 and set the value of NumberFound to 0
Step 3:  While ($i \le 10{,}000$) do steps 4 through 7
Step 4:      If NUMBER equals $T_i$ then
Step 5:      Print the name of the corresponding person, $N_i$
Step 6:      Set the value of NumberFound to NumberFound + 1
Step 7:      Add 1 to the value of $i$
Step 8:  Print the message NUMBER ' was found ' NumberFound  'times'
Step 9:  Stop
```

14. Let's assume that FindLargest is now a primitive to us, and use it to repeatedly remove the largest element from the list until we reach the median.

```
Step 1:  Get the values $L_1, L_2, \ldots, L_N$ of the numbers in the list
Step 2:  If $N$ is even then
             Let $M = N / 2$
         Else
             Let $M = (N + 1) / 2$
Step 3:  While ($N \ge M$) do steps 4 through 9
Step 4:      Use FindLargest to find the location of the largest number
              in the list $L_1, L_2, \ldots, L_N$
Step 5:      Exchange $L_{location}$ and $L_N$ as follows
Step 6:      $Temp = L_N$
Step 7:      $L_N = L_{location}$
Step 8:      $L_{location} = Temp$
Step 9:      Set $N$ to $N - 1$ and effectively shorten the list
Step 10:  Print the message 'The median is: '
Step 11:  Print the value of $L_M$
Step 12:  Stop
```

15. This algorithm will find the first occurrence of the largest element in the collection. This element will become *LargestSoFar*, and from then on A_i will be tested to see if it is greater than *LargestSoFar*. Some of the other elements are equal to *LargestSoFar* but none are greater than it.

16. (a) If $n \leq 2$, then the test would be true, so the loop would be executed. In fact, the test would never become false. Thus the algorithm would either loop forever, or generate an error when referring to an invalid A_i value. If $n > 2$, then the test would be false the first time through, so the loop would be skipped and A_1 would be reported as the largest value.

(b) The algorithm would find the largest of the first $n - 1$ elements and would not look at the last element, as the loop would exit when $i = n$.

(c) For $n = 2$ the loop would execute once, comparing the A_1 and A_2 values. Then the loop would quit on the next pass, returning the larger of the first two values. For any other value of n, the loop would be skipped, reporting A_1 as the largest value.

17. (a) The algorithm would still find the largest element in the list, but if the largest were not unique then the algorithm would find the last occurrence of the largest element in the list.

(b) The algorithm would find the smallest element in the list.

The relational operations are very important, and care must be taken to choose the correct one, for mixing them up can drastically change the output of the algorithm.

18. (a) The algorithm will find the three occurrences of "and". First in the word band, second in the word and, and third in the word handle.

(b) We could search for “ and ”. That is, the word itself surrounded by spaces. Note that the word "and" is special in that it is almost always surrounded by spaces in a sentence. Other words may start or end sentences and be followed by punctuation.

19. It would go into an infinite loop, because k will stay at 1, and we will never leave the outside while loop. We will keep checking the 1 position over and over again.

20.
```
Step 1:  Get the value for $N$
Step 2:  Set the value of $i$ to 2
Step 3:  Set the value of $R$ to 1;
Step 4:  While ($i < N$ and $R \neq 0$) do Steps 5-6
Step 5:      Set $R$ to the remainder upon computing $N/i$
Step 6:      Set the value of $i$ to $i + 1$
Step 7:  If $R = 0$ then
```

```
            Print the message 'not prime'
        Else
            Print the message 'prime'
```

(This algorithm could be improved upon because it is enough to look for divisors of N less than or equal to $\sqrt{N}$.)

21. Here we assume that we can perform "arithmetic" on characters, so that m + 3 = p, for example. Step 4 is the difficult part that must handle the "wraparound" from the end of the alphabet back to the beginning.

```
Step 1:  Get the values for nextChar and k
Step 2:  While (nextChar ≠ $) do steps 3 through 5
Step 3:      Set the value of outChar to nextChar + k
Step 4:      If outChar > z then
                 Set the value of outChar to (outChar -26)
Step 5:      Print outChar
```

22.

```
Step 1:  Get the values for k and $N_1$, $N_2$, …, $N_k$
Step 2:  Set the value of front to 1
Step 3:  Set the value of back to k
Step 4:  While (front ≤ back) do steps 5 through 9
Step 5:      Set the value of Temp to $N_{back}$
Step 6:      Set the value of $N_{back}$ to $N_{front}$
Step 7:      Set the value of $N_{front}$ to Temp
Step 8:      Set front = front + 1
Step 9       Set back = back – 1
```

23.

```
Step 1: Get the values for $N_1$, $N_2$, …, $N_k$, and SUM
Step 2:  Set the value of i to 1
Step 3:  Set the value of j to 2
Step 4:  While ($i < k$) do steps 5 through 11
Step 5:      While ($j \le k$) do steps 6 through 9
Step 6:          If $N_i + N_j = SUM$ then
Step 7:              Print ($N_i$, $N_j$)
Step 8:              Stop
                 Else
Step 9:              Set the value of $j$ to $j + 1$
Step 10:     Set the value of $i$ to $i + 1$
Step 11:     Set the value of $j$ to $i + 1$
Step 12:  Print the message 'Sorry, there is no such pair of values.'
```

24.

```
Set count to 0
Set sum to 0
Get a value for V
While V ≠ -1
    Set sum to sum + V
    Set count to count + 1
    Get the next value for V
End of the loop
Let's make sure that we had at least one value so we don't divide by 0
If (count > 0)
    Set average to sum / count
    Print the value of average
Else
    Print the message 'I was given no input data'
Stop
```

25.
```
Set adjacent to NO
Get values for V1 and V2     *We can do this since we know there are at least 2 values*
While (V2 ≠ -1) AND (adjacent = NO)
    If V1 = V2
        Set adjacent to YES
    Else
        Set V1 = V2
        Get a new value for V2
End of loop
Print the value of adjacent
Stop
```

26. We only need to make one simple change. Instead of writing

```
Print the value of adjacent
```

we change that to read:

```
If (adjacent = YES)
    Print the message 'Yes, the numbers ' V1 ' and ' V1 ' are adjacent.'
Else
    Print just the value of adjacent
```

Discussion of Challenge Work

1. The general algorithm is fairly clear, in English, in the text.

```
Step 1:  Read values for start, step, and accuracy
Step 2:  While |step| > accuracy  do steps 3 through 9
Step 3:        If f(start) > 0 then set FirstSign to +
Step 4:        Else set FirstSign to –
Step 5:         Do   steps 6 through 8
Step 6:              Set the value of start to start + step
Step 7:              If f(start) > 0 then set the value of Sign to +
Step 8:              Else set the value of Sign to –
                while  (Sign = FirstSign)
Step 9:         If |step| ≥ accuracy then set the value of step to (-0.1)step
Step 10:  Set the value of root to start – step/2
Step 11:  Print the value of root.
```

2. Many excellent simulations of sorting algorithms are available on the Web, suggest students examine them if they have questions about this algorithm.

The Find Largest algorithm given in the book always searches the whole list. First, we should create a variation that takes, in addition to the list of values, two indices which bound the range of the list that should be searched. Also, it is easiest to return the location of the largest value, for use in the sort algorithm. Below is a sketch of how it should change:

```
FindLargest(A, start, end)

    Step 1:  Set the value of loc to start
    Step 2:  Set the value of i to start + 1
    Step 3:  While (i ≤ end) do
    Step 4:       If $A_i > A_{loc}$ then
    Step 5:            Set loc to i
    Step 6:       Add 1 to the value of i
    Step 7:  End of the loop
    Step 8:  Return the value loc
```

The Selection Sort algorithm is quite simple, once we have a suitable form for the Find Largest portion of it.

```
    Step 1:  SelectionSort(A, n)
    Step 2:  Set lastpos to n
```

```
Step 3:  While ($lastpos \geq 1$) do
Step 4:      Set $biggestpos$ to FindLargest($A$, 1, $lastpos$)
Step 5:      Swap $A_{lastpos}$ and $A_{biggestpos}$
Step 6:      Subtract 1 from $lastpos$
Step 7:  End of loop
```

3. Students should be provided with concrete leads to reference materials about non-European mathematicians, including references to online resources.

Chapter 3: The Efficiency of Algorithms

1. (a) Group into 25 pairs of the form

$$2 + 100 = 102$$

$$4 + 98 = 102$$

.

.

.

$$50 + 52 = 102$$

so that the sum = 25(102) = 2550.

(b) $(n/2)(2n + 2) = n(n + 1)$

2. (a) Using Gauss's formula for the sum of the numbers from 1 to n, $\left(\frac{n}{2}\right)(n+1)$,

$$1 + 2 + 3 + \ldots + 12 = \left(\frac{12}{2}\right)(13) = 78$$

(b) Applying Gauss's formula for each of the 12 days, the total equals

$$\left(\frac{1}{2}\right)(2) + \left(\frac{2}{2}\right)(3) + \left(\frac{3}{2}\right)(4) + \cdots + \left(\frac{12}{2}\right)(13)$$

$$= \left(\frac{1}{2}\right)\left[1(2) + 2(3) + 3(4) + \cdots 12(13)\right]$$

$$= \frac{1}{2}\left(\frac{(12)(13)(14)}{3}\right) = \frac{2184}{6} = 364$$

3. (a) The terms of the Fibonacci sequence are:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765

so F(20) = 6765

(b) 6765

(c) Using the definition is probably clearer, because it is not easy to see why the formula is true. However, adding up the 20 terms in the definition to find *F*(20) is somewhat less efficient. For *F*(100), the formula is definitely quicker.

(d) Once cell C1 contains =A1 + B1, which computes F(3),you can drag this formula along line 1 as far as you want so that cell D1 computes the result B1 + C1 = F(4), cell E1 computes the result C1 + D1 = F(5), etc. The value of F(20) is displayed in the 20th cell of row 1, which is cell T1.

(e) Assuming the value of n is stored in cell B6, the formula to compute F(n) is

(SQRT(5)/5)*POWER((1 +SQRT(5))/2,B6)-(SQRT(5)/5)*POWER((1 - SQRT(5))/2,B6)

(f) To compute F(50) using the definition, you have to drag the formula to cell AX1. T0 compute F(50) using the definition, you just have to enter 50 in cell B6. Even though the spreadsheet is doing all the work, it still seems quicker to just change the value of n. In either case, F(50) = 12,586,269,025

4. (a) 171 + 85 + 43 + 21 + 11 + 5 + 3 + 1 + 1 = 341

(b) There is 1 winner so there must be 341 losers, therefore 341 matches.

(c) The second algorithm is more elegant and efficient. Perhaps the first is clearer, since the second is so clever.

5. (a) middle position = 8.

8 comparisons required.

(*n* + 1)/2.

(b) two middle positions = 8 and 9.

8 and 9 comparisons required, average = 8.5.

(*n* + 1)/2.

(c) (*n* + 1)/2.

6. 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28.

28/7 = 4.

Yes, the number is slightly higher than *n*/2 = 7/2 = 3.5, and it is exactly (*n* + 1)/2 = 8/2 = 4 (see Exercise 5).

7. (a) (32,000,000/2) × (1/12000) = 1333.3 secs = more than 22 minutes

(b) $2^{25} \approx 33{,}000{,}000$ so lg (32,000,000) = 25. Therefore the time required would be

$$25 \times (1/12000) = 0.002 \text{ secs}$$

8. At 14 pancakes, both formulas give the same result (25 flips). If the number of pancakes is more than 14, the new algorithm is faster, but may give a fractional answer, which does not make physical sense. At 17 pancakes, the original algorithm requires 31 flips and the new formula requires 30 flips.

9. 7, 4, 2, 9, 6 (original list)

7, 4, 2, 6, 9

6, 4, 2, 7, 9

2, 4, 6, 7, 9

10. When the unsorted section consists of one number, the largest so far is set to that number. There are no other numbers to compare it to so the find-largest algorithm terminates, leaving that number as the largest. This process uses zero comparisons, so adds nothing to the total of the comparisons done, and removing it has no effect.

11. (a)

4 8 2 6

4 **2 8** 6

4 2 **6 8**

2 4 6 8

(b)

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

(c)

D B G F A C E H

B **D** G F A C E H

B D **F** **G** A C E H

B D F **A** **G** C E H

B D F A **C** **G** E H

B D F A C **E** **G** **H**

B D **A** **F** C E G H

B D A **C** **F** E G H

B D A C **E** **F** G H

B **A** **D** C E F G H

B A **C** **D** E F G H

A **B** C D E F G H

For these cases, bubble sort required the same or more exchanges than selection sort.

12. Initially, the entire list is unsorted (the marker *U* for the unsorted section is at the end of the list, position *n*). Within the loop at step 3, the marker is decreased (at step 8) once for each pass until the unsorted section of the list is reduced to one element, which means *U* = 1. The inner loop at step 5 is always done as many times as it takes to move C through positions 2, … , U.

Value of U	Number of passes through loop at step 5	
n	n - 1	(C goes from 2 through n)
$n – 1$	$n – 2$	(C goes from 2 through n - 1)
. . .	. . .	. . .
2	1	(C goes from 2 through 2)

The loop at step 5 is therefore done $(n – 1) + (n – 2) + . . . + 2 + 1$ times, which is $\Theta(n^2)$. Each pass through this loop requires one comparison, at step 6.

13. Selection sort does one exchange for each position in the list, regardless of what the list looks like. Bubble sort only exchanges pairs that are out of order, so if the list is already sorted, it will do no exchanges.

14. (a)

```
1.  Get values for n and the n list items
2.  Set the marker U for the unsorted section at the end of the list
    Set the value of Exchanges to 1
3.  While (Exchanges > 0) and the unsorted section has more than one element do
     steps 4 through 11
4.      Set the current element marker C at the second element of the list
5.      Set the value of Exchanges to 0
6.      While C is has not passed U do steps 7-10
7.          If the item at position C is less than the item to its left then do steps 8 and 9
8.              Exchange these two items
9.              Add 1 to the value of Exchanges
10.         Move C to the right one position
11. Move U left one position
12. Stop
```

(b)

7, 4, 12, 9, 11 |	7 and 4 are compared (and exchanged)
4, 7, 12, 9, 11 |	7 and 12 are compared
4, 7, 12, 9, 11 |	12 and 9 are compared (and exchanged)
4, 7, 9, 12, 11 |	12 and 11 are compared (and exchanged)
4, 7, 9, 11, 12	

End of Pass 1, 4 comparisons (and 3 exchanges) took place

4, 7, 9, 11 | 12	4 and 7 are compared
4, 7, 9, 11 | 12	7 and 9 are compared
4, 7, 9, 11 | 12	9 and 11 are compared

End of Pass 2, 3 comparisons (and 0 exchanges) took place

The algorithm stops; a total of 7 comparisons was done.

(c) An already sorted list is the best case; it requires n – 1 comparisons and 0 exchanges.

(d) In a reverse-sorted list, smart bubble sort acts exactly the same as regular bubble sort.

15.

A = 8, 12, 19, 34	B = 3, 5, 15, 21	C =
A = 8, 12, 19, 34	B = 5, 15, 21	C = 3
A = 8, 12, 19, 34	B = 15, 21	C = 3, 5
A = 12, 19, 34	B = 15, 21	C = 3, 5, 8
A = 19, 34	B = 15, 21	C = 3, 5, 8, 12
A = 19, 34	B = 21	C = 3, 5, 8, 12, 15
A = 34	B = 21	C = 3, 5, 8, 12, 15, 19
A = 34	B =	C = 3, 5, 8, 12, 15, 19, 21
A =	B =	C = 3, 5, 8, 12, 15, 19, 21, 34

16. The original list is 6, 3, 1, 9, with size > 1. Split the list into two halves (step 3):
6, 3   1, 9

Use mergesort on the first half, 6, 3, with size > 1. Split the list into two halves (step 3):
6   3
Each half contains 1 item, so there is nothing to do. This completes steps 4 and 5 for the list 6, 3, so step 6 requires merging A = 6 and B = 3 to get C = 3, 6 (2 comparisons). This completes step 6 for the first half, so mergesort on the first half stops (step 7).

Use mergesort on the second half, 1, 9, with size > 1 Split the list into two halves (step 3):

1 9

Each half contains 1 item, so there is nothing to do. This completes steps 4 and 5 for the list 1, 9, so step 6 requires merging A = 1 and B = 9 to get C = 1, 9 (2 comparisons). This completes step 6 for the second half, so mergesort on the second half stops (step 7).

This completes steps 4 and 5 on the list 6, 3, 1, 9. Now merge A = 3, 6, B = 1, 9 to get C = 1, 3, 6, 9 (4 comparisons).

This completes step 6 on the list 6, 3, 1, 9, so the algorithm stops, with the resulting sorted list 1, 3, 6, 9 that required a total of 8 comparisons.

17. (a) lg n

(b) n

(c) Θ(n lg n)

(d) Yes. Exercise 16 required 8 comparisons, which equals 4*lg 4 = 4*2 = 8.

18. (a) 4, 3, 7

4, 7, 3

3, 4, 7

3, 7, 4

7, 4, 3

7, 3, 4

(b) 3! = (3)(2)(1) = 6. This agrees with the number of permutations from Part a.

(c) The body of the while loop that contains step 5 is always executed *n*! times. In the best case, the first permutation checked is the sorted permutation, so step 10 is executed only once, giving a total of *n*! + 1 work units. In the worst case, the last permutation checked is the sorted permutation, so step 10 is executed *n*! times, giving a total of 2*n*! work units.

(d)

Order	n = 3	5	10	20
n^2	0.0009 sec	0.0025 sec	0.01 sec	0.04 sec
$n!$	0.0006 sec	0.012 sec	6 min	77×10^3 centuries

(e) The new algorithm requires $n!$ copies of the original list, so it is extremely space-inefficient (as well as time-inefficient).

19. We want to find the value of n where the number of instructions executed by the two algorithms is equal. Above that point, algorithm B will be faster. So we must solve the equation:

$$0.003n^2 = 243n$$

to get n = 81,000

20. (a)
```
For each of rows 1 through $n$ do the following
    For each of columns 1 through $n$ do the following
        Print the entry in this row and column
```

n^2 print statements

(b)
```
For each of the districts 1 through $n$ do the following
    For each of rows 1 through $n$ do the following
        For each of columns 1 through $n$ do the following
            Print the entry in this row and column
```

n^3 print statements.

(c) $\Theta(n^3)$

21. legit = 6.

3 2 6 7 5 1 1 1 1 1

8 + 8 + 5 + 5 = 26 copies.

22. legit = 6.

3 1 5 2 6 7 0 0 5 1

3 copies.

23. 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 end have been copied; step 11 of the algorithm can be changed to “while right is less than or equal to legit do steps 12 and 13." 8 + 7 + 3 + 2 = 20 copies

24. 30 copies = 6(6 – 1).

25. (a) John, Elsa

(b) John, Lee, Snyder, Tracy

(c) John, Elsa, JoAnn

26. 14, 22, 31, 43

27.

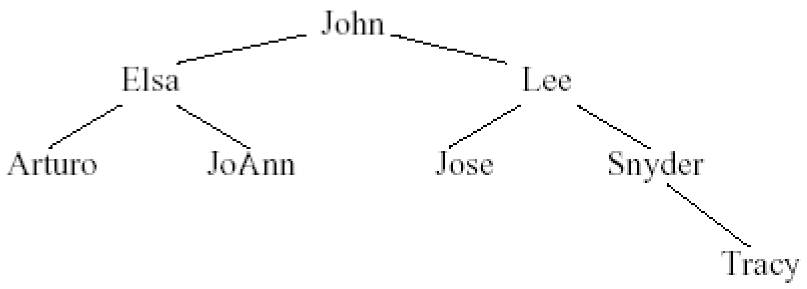
```
Step 1:  Get values for the target to be searched for, n, and the n list items
Step 2:  Set the value of i to 1, set the value of Found to NO, set the value of Greater to NO
Step 3:  While (Found = NO) and (i ≤ n) and (Greater = NO) do steps 4 through 8
Step 4:          If target is equal to the ith element then do steps 5 and 6
Step 5:              Set the value of Found to YES
Step 6:              Print the message "Item found at location" i
Step 7:          If element i is greater than target, set the value of Greater to YES
Step 8:          Add 1 to the value of i
Step 9:  If Found = NO, then print the message 'Sorry, the target is not in this list'
Step 10: Stop
```

28. (a) *n* (for example, if target is equal to last element in list, must check all *n* elements)

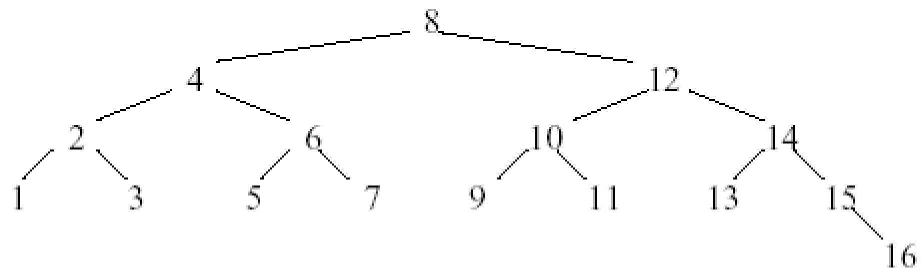
(b) *n*/2 – element will be found just as in regular sequential search

(c) Yes – for elements that are not in the list but are less than some of the list elements, this algorithm will terminate sooner than regular sequential search. (But remember that the list must already be sorted.)

29. Worst case = 4; Tracy.



30. Worst case = 5 for the value at position16.

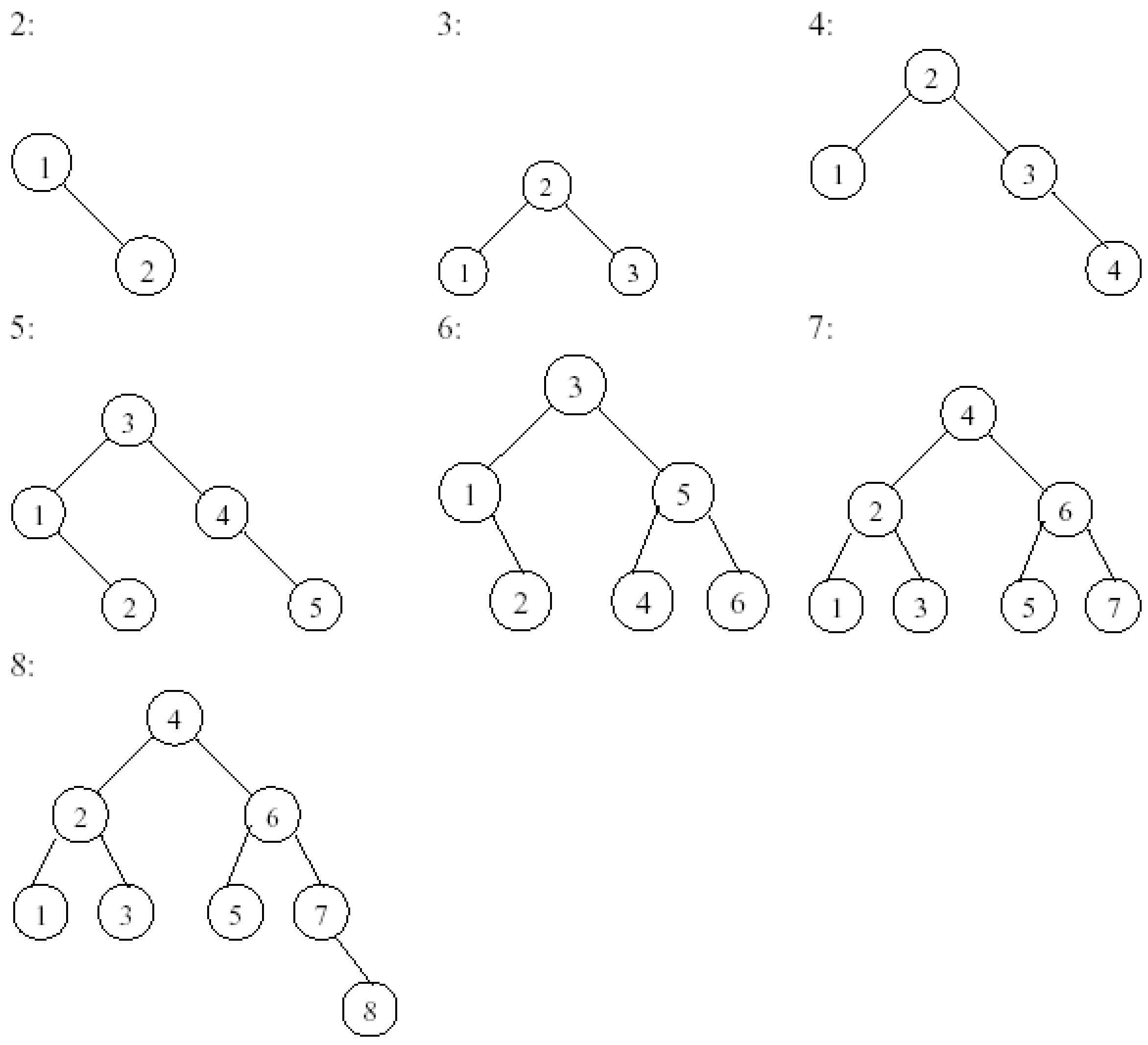


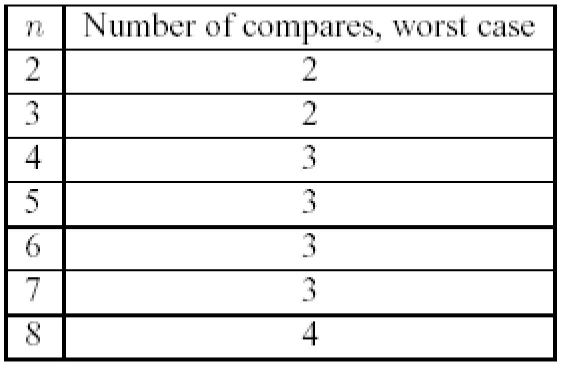
31. (a) 1, 2, 8, -5

(b)

n	$\lfloor \lg n \rfloor$
2	1
3	1
4	2
5	2
6	2
7	2
8	3

(c)





n	Number of compares, worst case
2	2
3	2
4	3
5	3
6	3
7	3
8	4

(d) $1 + \lfloor \lg n \rfloor$

32. Average = $(1 \times 1 + 2 \times 2 + 4 \times 3) / 7 = 17 / 7 = 2.43$. The worst case is 3, so this is somewhat less.

33. For sequential search, the worst case is $n = 100{,}000$, so for p searches, $p \times 100{,}000$ comparisons are required. Using an $\Theta(n^2)$ sorting algorithm would require $100{,}000 \times 100{,}000$ comparisons. Binary search would require $1 + \lfloor \lg n \rfloor = 1 + \lfloor \lg 100{,}000 \rfloor = 17$ comparisons for each search, so altogether there would be $100{,}000 \times 100{,}000 + 17p$ comparisons. The first expression is larger than the second for $p > 100{,}017$.

34. (a) The worst, as before, is if the pattern almost occurs everywhere in the text.

Text: AAAAAAAAA Pattern: AAAB

Number of comparisons: $m(n - m + 1)$

(b) The best case is if the pattern occurs as the first m characters of the text.

Text: ABCDEFGHI Pattern: ABC

Number of comparisons: m

35. (a) 6

(b) Each node in the graph has 4 choices for the next node in a path, so there are $4^6 = 4096$ paths.

(c) 4096 paths × 0.0001 seconds per path = 0.4096 sec

(d) 4^{12} paths × 0.0001 seconds per path equals about 28 minutes.

36. (a) The first graph has several Euler paths, for example, C-A-B-C-D-B

The second graph has no Euler path.

(b) (i) has 0 odd nodes, so an Euler path exists, for example A-C-B-D-A-E-B-F-A

(ii) has 4 odd nodes, so no Euler path exists

(iii) has 2 odd nodes, so an Euler path exists, for example C-A-E-B-F-A-D-C-B-D

(c) $\Theta(n^2)$

(d) No, a polynomial algorithm exists

37. Solve the equation

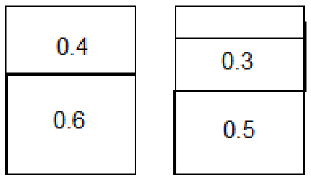
$$100n^2 = 0.01(2^n)$$

to get $n = 23$

38. (a) $5 \times 10 = 50$

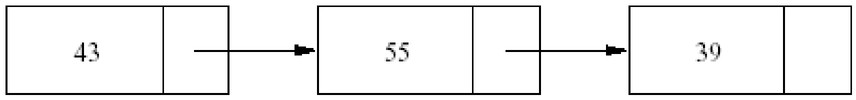
(b) $5^2 \times 10 = 25 \times 10 = 250$

39.

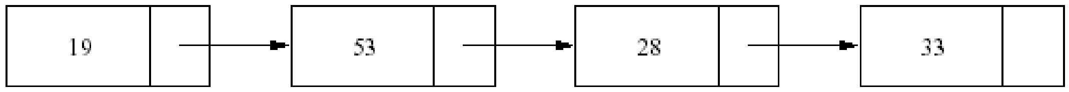


40.

(a)



(b)



(c) Locate the first item. Follow the pointers through the list; if the next list item ever has the value 0, then change the pointer of the current list item to point to where the zero item is pointing (removing the zero item). Stop at the end of the list.

There are at most $n – 1$ "follow pointer" pointer operations and at most $n – 2$ "change pointer" operations, hence this is an $\Theta(n)$ algorithm.

41. (a) 2, 1, 2, 1, 2, 1, 2, 1

(b) 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1, 4, 2, 1

(c) $\Theta(n \lg n)$

(d) $16 \times 4 = 64$

42. (a) n - 1 (*largest so far* must be compared with all elements in the list after item 1)

(b) $\Theta(n^2)$ to sort + 0 comparisons required to get the second largest value = $\Theta(n^2)$

(c) $(n - 1) + (n - 2) = \Theta(n)$

Discussion of Challenge Work

1. (a) There are 37 syllables in each verse that are not dependent on which verse it is (from the first two and the last lines of each verse). So clearly after n verses, there are $37n$ syllables from these lines. The remaining part of each verse is about each animal's particular noise. The lines for each animal take 22 syllables, and there are $1 + 2 + . . . + n$ copies of those lines of the song overall (one copy for the first verse, two for the second verse, and so on). So that means there are $22(1 + 2 + . . . + n)$ additional syllables. But $1 + 2 + . . . + n$ is just $n(n + 1) / 2$, so the total number of syllables is $22n(n + 1)/2 + 37n$.

   (b) This means that the song is $\Theta(n^2)$ since $n(n + 1)/2$ generalizes to n^2.

2. Students should research the simplex method in more detail than this question asks in order to fully understand it.

Chapter 4: The Building Blocks: Binary Numbers, Boolean Logic, and Gates

1. (a) $1 \times 4^2 + 3 \times 4 + 3 = 31$

(b) $3 \times 8^2 + 6 \times 8 + 7 = 247$

(c) $1 \times 16^2 + 11 \times 16 + 10 = 442$

2. There are only 10 available decimal digits, 0 . . . 9. To write a number in hexadecimal requires 16 digits to represent 0 . . . 15, so 0 . . . 9, A, B, . . . F are used. Notice that in hexadecimal the number 10 represents decimal 16, not decimal 10.

3. (a) 24; (b) 49: (c) 127; (d) 512

4. (a) 23 = 00010111

(b) 55 = 00110111

(c) 275 = 100010011 but this exceeds 8 bits, resulting in an arithmetic overflow.

5. (a) -49; (b) 408; (c) -1; (d) 0

6. (a) 01000111 (71 = 64 + 4 + 2 + 1)
(b) 10000001
(c) 11010001 (81 = 64 + 16 + 1)

7. The magnitude 200 in binary is 11001000, which is 8 bits. However, since the sign takes up one bit we only have 7 left for the magnitude, which means that -200 is too big to represent in 8-bits. This would cause an *overflow error*.

8. 511

9.

```
            0   ← carry digit
   0011100011
 + 0001101110
 ------------
            1

           1    ← carry digit
   0011100011
 + 0001101110
 ------------
           01


          1     ← carry digit
   0011100011
 + 0001101110
 ------------
          001
```

$$\begin{array}{r l} 1 & \leftarrow \text{carry digit} \\ 0011100011 & \\ \underline{+\ \ 0001101110} & \\ 0001 & \end{array}$$

$$\begin{array}{r l} 0 & \leftarrow \text{carry digit} \\ 0011100011 & \\ \underline{+\ \ 0001101110} & \\ 10001 & \end{array}$$

$$\begin{array}{r l} 1 & \leftarrow \text{carry digit} \\ 0011100011 & \\ \underline{+\ \ 0001101110} & \\ 010001 & \end{array}$$

$$\begin{array}{r l} 1 & \leftarrow \text{carry digit} \\ 0011100011 & \\ \underline{+\ \ 0001101110} & \\ 1010001 & \end{array}$$

$$\begin{array}{r l} 1 & \leftarrow \text{carry digit} \\ 0011100011 & \\ \underline{+\ \ 0001101110} & \\ 01010001 & \end{array}$$

$$\begin{array}{r l} 0 & \leftarrow \text{carry digit} \\ 0011100011 & \\ \underline{+\ \ 0001101110} & \\ 101010001 & \end{array}$$

$$\begin{array}{r} 0011100011 \\ \underline{+\ \ 0001101110} \\ 0101010001 \end{array}$$

10.

	Dec. Number	Bin. Number	Internal Representation			
(a)	$+7.5$	$+111.1$	0	111100000	0	00011
(b)	-20.25	-10100.01	1	101000100	0	00101
(c)	$-1/64$	-0.000001	1	100000000	1	00101

11. (a) The mantissa is 0.111000000, which is +(1/2 + 1/4 + 1/8) = +7/8
The exponent is 0 00111 which is + (1 + 2 + 4) = +7
So, the overall value is +7/8 × 2^{+7} = +7/8 × 128 = +112.0

(b) The mantissa is 1.010001000 = -(1/4 + 1/64) = -(17/64)
The exponent is 1 00001 which is -1
So, the overall value is $(-17/64) \times 2^{-1} = (-17/64) \times 1/2 = -17/128 = -0.132813$
(Note: Although this is the correct answer, the mantissa is not normalized. In most computers this value would be normalized so the first digit of the mantissa is a 1, and the exponent would be adjusted accordingly.)

12. If we gave more bits to the mantissa we could represent our mantissa more accurately, resulting in more significant digits. However, since we have fewer digits for the exponent, we cannot represent exponents as large as before, resulting in an inability to represent larger (or smaller, if the exponent is negative) numbers. Essentially, we will get increased precision at the price of a decreased range.

13. (a) 01000001 01100010 01000011

(b) 01001101 01101001 01101011 01100101

(c) 00100100 00110010 00110101 00101110 00110000 00110000

(d) 00101000 01100001 00100000 00101011 00100000 01100010 00101001

14. There are 30 characters in "Invitation to Computer Science", excluding the quotation marks. In ASCII, each character takes 8 bits, for a total of 240 bits. Each character in UNICODE takes 16 bits, for a total of 480 bits.

15. **(a)** A 3-minute song has 180 seconds in it. Each second is sampled 40,000 times so there are 7,200,000 samples. Each sample is stored using 16 bits, thus the total number of bits (uncompressed) is 115,200,000. A compression scheme with a 5:1 ratio would reduce the size to 23,040,000 bits.

(b) RGB-color images use 3 bytes per pixel (one for each of the three color contributions), a total of 24 bits per pixel. A 1,200 by 800 image has 960,000 pixels, so in total it would use 23,040,000 bits. If it uses 2.4 Mbits, that means it uses 2,400,000 bits, so the compression ratio would be 9.6, or essentially 10:1.

16. Run-length encoding of xxxyyyyyyzzzzAAxxxx would be (x,3) (y,6) (z,4) (A,2) (x,4). Assuming one byte per character, the original takes 19 bytes, and the encoding takes 10 bytes, for a compression ratio of 19:10, almost 2:1.

17. (a) The word KAI would be 11101 00 10 in the variable length code given. It takes 9 bits, compared to 12 bits in the 4-bit encoding. The savings is 4:3, or about 1.33.

(b) MAUI would be 11100 00 11110 10 in the variable length code. It takes 14 bits, compared to 16 bits in the 4-bit encoding. The savings is 8:7, or about 1.14.

(c) MOLOKAI would be 11100 0111 1111111 0111 11101 00 10 in the variable length code. It takes 29 bits, compared to 28 bits in the 4-bit encoding. In this case, the variable length code is actually longer, because it contains many less-frequent letters. The "savings" is 28:29, or 0.97.

18. You can store fewer numbers in binary with the same number of places than you can in decimal. For example with 8 digits in binary you can store the numbers 0 to 255, and in decimal you can store 0 to 99999999. Other answers include the fact that the designers of the computer hardware must work in binary, which is not a familiar numbering system, and the fact that translation must be done between the external representation of numbers (in decimal) and the internal representation of numbers (in binary).

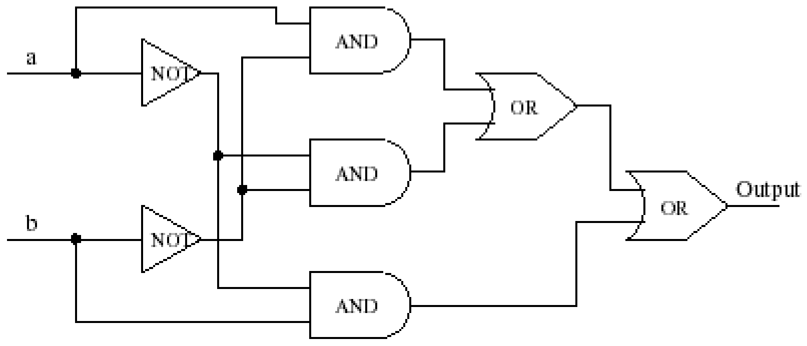
19. (a) True; (b) False; (c) False; (d) False; (e) False.

20. The order of operations is unclear. You can use parenthesis to indicate which is to be done first, or set the rules of precedence. Usually, AND is considered like multiplication, and takes precedence over OR, so we would consider the expression equivalent to ((a = 1) AND (b = 2)) OR (c = 3).

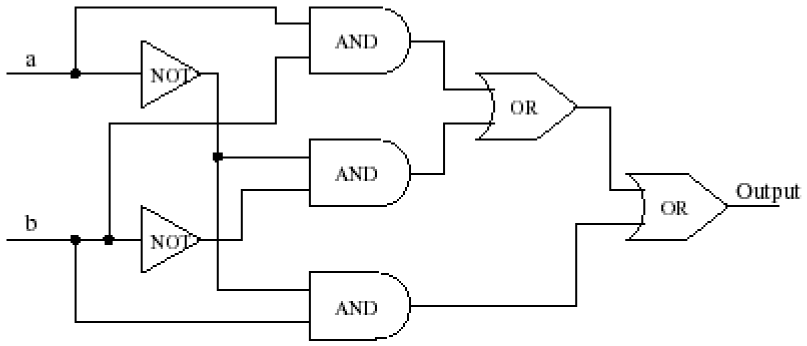
21. A truth table with k variables will always have 2^k rows. So, when k = 5, our truth table will have $2^5 = 32$ rows.

Note: In all circuit diagrams we assume the existence of 3-input AND gates for simplicity.

22. The circuit corresponds to the Boolean expression $(\overline{a} \cdot \overline{b}) + (\overline{a} \cdot b) + (a \cdot \overline{b})$.



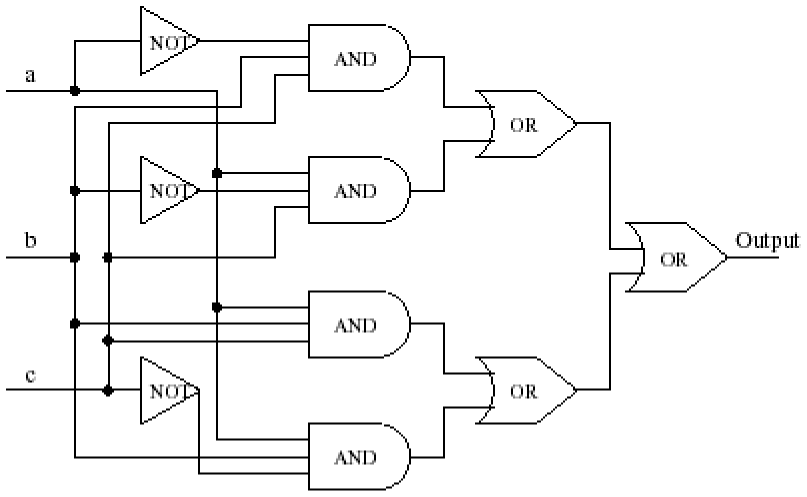
23. The circuit corresponds to $(\overline{a} \cdot \overline{b}) + (\overline{a} \cdot b) + (a \cdot b)$.



24. The truth table corresponding to this circuit is as follows.

a	b	c	$Output$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

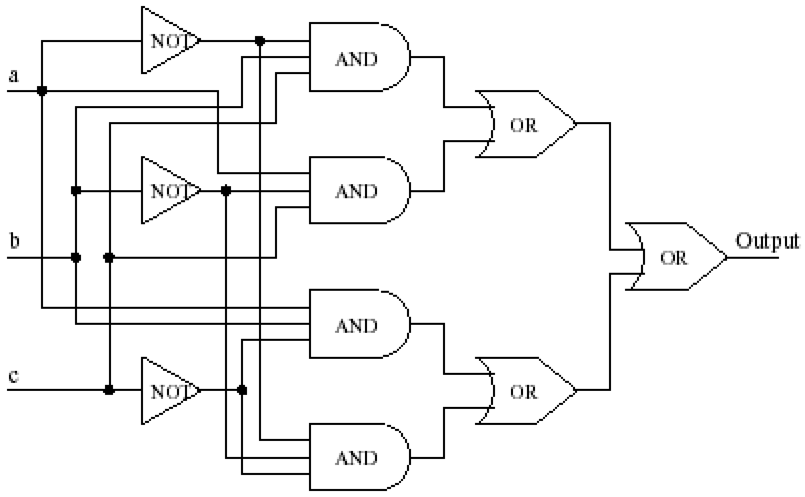
The circuit corresponds to $(\overline{a} \cdot b \cdot c) + (a \cdot \overline{b} \cdot c) + (a \cdot b \cdot \overline{c}) + (a \cdot b \cdot c)$.



25. The truth table corresponding to this circuit is as follows:

a	b	c	$Output$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

The circuit corresponds to $(\overline{a} \cdot \overline{b} \cdot \overline{c}) + (\overline{a} \cdot b \cdot c) + (a \cdot \overline{b} \cdot c) + (a \cdot b \cdot \overline{c})$.



26. This is a difficult problem that will challenge the best students in the class. You have to do several examples to determine the truth table. The truth table is:

Previous Borrow (PB)	a	b	$a - b$	Next Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

So the sum of products expressions for *a – b* and Next Borrow are

$a - b = (\overline{PB} \cdot a \cdot \overline{b}) + (\overline{PB} \cdot \overline{a} \cdot b) + (PB \cdot a \cdot b) + (PB \cdot \overline{a} \cdot \overline{b})$

Next Borrow = $(\overline{PB} \cdot \overline{a} \cdot b) + (PB \cdot a \cdot b) + (PB \cdot \overline{a} \cdot b) + (PB \cdot \overline{a} \cdot \overline{b})$

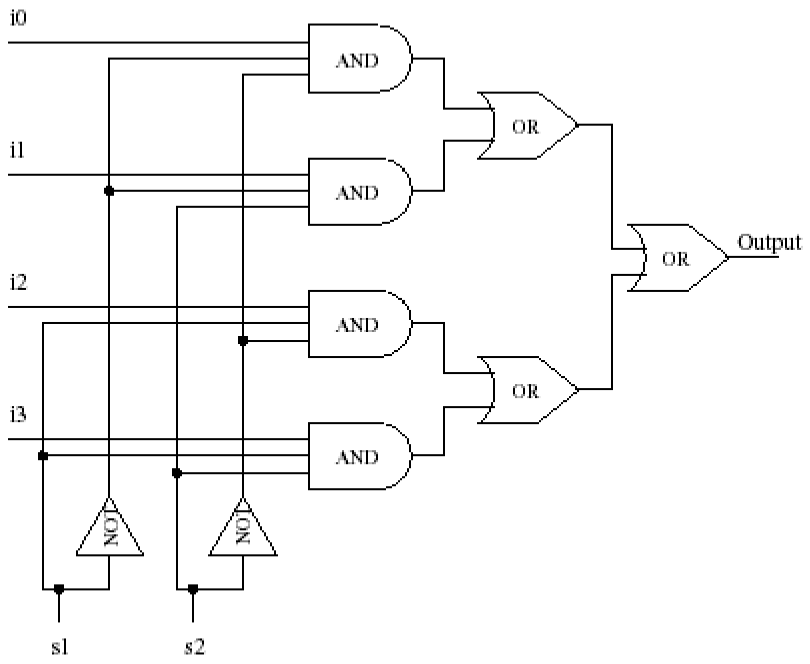
The circuit may be constructed as in the previous problems.

27. A multiplexor with 2^N input lines has N selector lines, so a four-input multiplexor will need 2 input lines and an eight-input multiplexor will need 3 input lines.

28. Suppose the input lines are labeled as i_0, i_1, i_2, and i_3 and the selector lines are labeled (left to right) as s_1 and s_2. i_0 will be the output if both s_1 and s_2 are zero, i_1 will be the output if $s_1 = 0$ and $s_2 = 1$, and so forth. The Boolean expression for a 4-input multiplexor is therefore

$$(i_0 \cdot \overline{s_1} \cdot \overline{s_2}) + (i_1 \cdot \overline{s_1} \cdot s_2) + (i_2 \cdot s_1 \cdot \overline{s_2}) + (i_3 \cdot s_1 \cdot s_2).$$

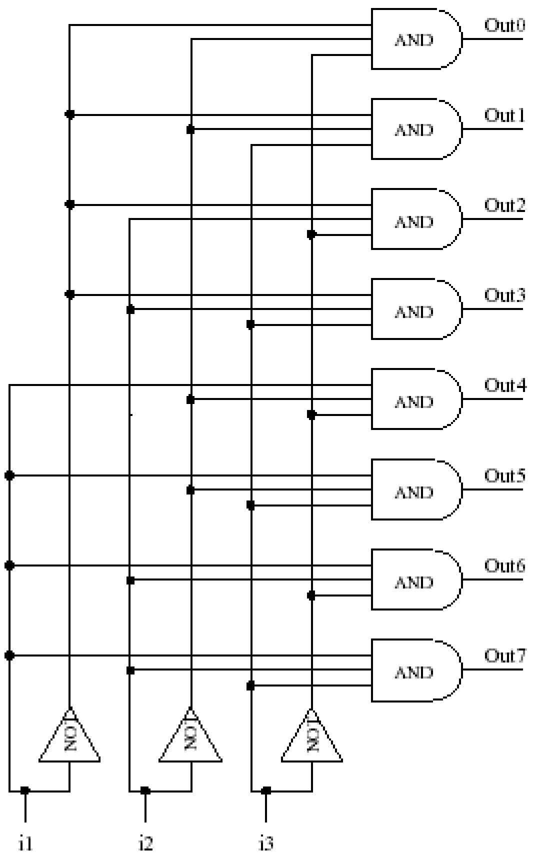
The circuit is



29. Suppose the input lines are labeled as i_1, i_2, and i_3, and the outputs are labeled Out_0 through Out_7. Out_0 is turned on when i_1, i_2, and i_3 are all zero, Out_3 is on when $i_1 = 0$, $i_2 = 1$, and $i_3 = 1$, and so forth. The circuit must have 8 subparts, one for each possible combination of input values. Each part ANDs together a particular Boolean expression like

$$\overline{i_1} \cdot i_2 \cdot i_3$$

and the result is the corresponding output line (*Out*3, in this case).



Discussion of Challenge Work

1. Changes may really only be made to the 1-bit adder. However a small change in the 1-bit adder will multiply to a larger effect in the overall circuit. An easy first step is to have both the s_i and c_{i-1} outputs share the portion of the circuits that computes $a \cdot b \cdot c$. There is no other reason to have two copies of that sub-circuit. That alone will save two AND gates per 1-bit adder. For a 32-bit adder, it will save 64 × 3 transistors. As an example of further simplifications, we can simplify $(\bar{a} \cdot \bar{b} \cdot c) + (\bar{a} \cdot b \cdot \bar{c})$ in the s_i calculation to be $\bar{a} \cdot ((\bar{b} \cdot c) + (b \cdot \bar{c}))$ which takes 4 gates instead of 5. This will save 3 transistors per 1-bit adder, for a total of 96 transistors saved overall. Similar simplifications may be done.

2. Students will need a good introductory resource for two's complement, but otherwise it should be a straightforward problem.

3. Given that jpeg images are often found on the Web, students should find this an interesting topic.

Chapter 5: Computer Systems Organization

1. The advantage of using a very large memory cell size is that the computer can store larger (in terms of absolute value) values in its memory cells. The disadvantage is that there would be fewer cells available, and with such big cells there would be a lot of wasted space when storing small numbers. The largest positive integer that could be stored in a system using sign/magnitude notation, with 64-bit cells is 2^{63}. If two such cells were used to store integers, the largest integer that could be stored by the system is 2^{127}.

2. a. 20 bits (2^{20} = 1,048,576, just over one million)
b. 24 bits
c. 27 bits
d. 30 bits

3. A memory unit with 640 KB has 640×2^{10} memory cells, provided that the computer uses 8-bit cells. A memory unit with 512 MB would contain 512×2^{20} memory cells. A memory unit with 2 GB would contain 2×2^{30} cells.

4. Read-only memory can be used to store essential system instructions that users cannot overwrite. The information could originally be stored in the ROM by hardwiring the electronic circuitry.

5. The dimensions of a memory containing 1 MB of storage would be 1024 × 1024. The MAR would have to be 20 bits large. 10 bits would be sent to each of the row and column decoders, and the decoders would each have 1024 output lines.

6. The maximum size of the memory unit on this machine is 2^{24}. The dimensions of the two-dimensional memory are $2^{12} \times 2^{12}$ or 4096 × 4096.

7. One could store 4 MB of memory using a 20 bit MAR by thinking of our 4 MB memory as being composed of four separate 1 MB units. Then if we had a separate 2-bit register that specified which of the four units we were accessing, we could use the 20 bit MAR to access into that one particular memory unit. In a sense, the new 2-bit register could be thought of as the two high-order bits of a 22-bit MAR.

8. We would have to make two fetches to get the complete instruction. Alternately, we might fetch only the first 16 bits, which could be the operation code, and then decode that part of the instruction. Then, during the execution phase, we might go back to memory to fetch the last 16 bits, which could be the address of the operand on which we are going to work. Basically, we would have to either extend the fetch phase or do part of the fetch in the execution cycle.

9. One page = 65 × 60 = 3900 characters.

$$\frac{\text{3900 characters}}{\text{780 characters/second}} = \text{5 seconds}$$

A 1 gigaflop machine does 1,000,000,000 floating point operations per second, so in that time it can perform

$5 \times 1{,}000{,}000{,}000 = 5{,}000{,}000{,}000$ floating point operations

10. One would need a multiplexor circuit that contains 5 selector lines because $2^5 = 32$ is sufficient to identify one of 20 lines. With only 4 selector lines, we would not have enough selectivity because $2^4 = 16$.

11. Major advantages: 1) Since each instruction does a lot more, the size of the programs will be less, and 2) the programmers will have an easier time since they are working with more sophisticated and powerful instructions.
Major disadvantage: The circuitry will be much bigger and more complex, taking up far more room on the chip. This will leave much less room to do things like optimization.

12. a. The number of characters per disk = 1024 char/sector × 20 sectors/track × 500 tracks/surface × 2 surfaces = 20,480,000 characters.

b. 7200 rev/min = 120 rev/sec or 1/120 = 0.00833 sec/rev = 8.33 msec/rev.
20 sectors per track = 20 sectors/rev or 1/20 = 0.05 rev/sector.
Finally,

8.33 msec/rev × 0.05 rev/sector = 0.4165 msec/sector

This is the transfer rate, which is a constant.

	Best	Worst	Average
Seek Time	0	0.5 + 499 × 0.05 = 25.45	0.5 + 150 × 0.05 = 8 (assume average move is 150 tracks)
Latency	0	8.33 (one complete revolution)	4.165 (one half revolution)
Transfer	0.4165	0.4165	0.4165
Total	0.4165	34.1965	12.5815

13. Since we are changing the rotational speed, the latency and transfer times will change but the seek times remain the same.

9600 rev/min = 160 rev/sec or 1/160 = 0.00625 sec/rev = 6.25 msec/rev.
6.25 msec/rev × 0.05 rev/sector = 0.3125 msec/sector (constant transfer rate)

	Best	Worst	Average
Seek Time	0.	0.5 + 499 × 0.05 = 25.45	0.5 + 150 × 0.05 = 8 (assume average move is 150 tracks)
Latency	0	6.25 (one complete revolution)	3.125 (one half revolution)
Transfer	0.3125	0.3125	0.3125
Total	0.3125	32.0125	11.4375

14 How can we minimize access time to this data? Latency time and transfer time cannot be improved but we can reduce the time that we spend moving the disk arm to the correct track. To do this we should store the information to first fill one track and then the corresponding track on the other side of the disk. This will store 20 × 1024 × 2 = 40,960 bytes. Store the remaining information on an adjacent track. This will minimize seek time to access this information once the initial track is located.

15. Having a read/write head per track eliminates seek time in all cases.

16. If we assume that the arm is initially positioned at the beginning of the first sector of the first track on the disk then we would have to perform the following operations:

a. read the entire first track on both sides. Since the disk is double sided, we will need to spend two revolutions to read each track—the first revolution to read the top surface and the second revolution to read the bottom.
b move the arm inward one track and wait for the beginning of the track
c. repeat steps a and b 499 more times to read every track on the disk.

Step a takes two revolutions, which is 2 × 8.33 msec = 16.66 msec
Step b takes 0.55 msec to move the arm, but this takes place while the disk continues to rotate, so the 0.55 msec is absorbed in the worst case of waiting for one complete revolution for the beginning of the track to rotate around under the head. Therefore step b takes a total of 8.33 msec.

So the total time will be 16.66 + 499 (8.33 + 16.66) = 12.48 seconds

17. A sequential access storage device such as a tape could be a useful form of mass storage when one is sequentially copying the memory of a disk drive to back up or archive a system.

18. 56000 bits are being transferred per second so the transfer time is 1/56000 sec/bit. The number of instructions executable between bits is

500,000,000 instructions/sec × 1/56000 sec/bit = 8928 instructions/bit

19. a. The maximum number of distinct operation codes that can be recognized and executed by the processor on this machine is $2^6 = 64$

b. The max memory size on this machine is 2^{18}

c. The number of bytes required for each operation is 6 (48 bits), since the IR is 6 + 18 + 18 = 42 bits.

20. With an op code field of 8 bits we could theoretically have $2^8 = 256$ operations.

21. For reference: v-200 w -201 x - 202 y -203 z-204

a. set v to x - y + z (assume command SUBTRACT X,Y,Z exists which is z = x - y)

Memory Location	*Op Code*	*Address Field*	*Comment*
50	SUBTRACT	202, 203, 200	v now has the value x - y
51	ADD	200, 204, 200	v now has the value x – y + z

b. set v to value of (w + x) - (y + z)

Memory Location	*Op Code*	*Address Field*	*Comment*
50	ADD	201, 202, 200	v now has the value w + x
51	ADD	203, 204, 201	w now has the value y + z
52	SUBTRACT	200, 201, 200	v now has the value (w + x) – (y + z)

c. if (v ≥ w) then

```
    set x to y
else
    set x to z
```

Memory Location	*Op Code*	*Address Field*	*Comment*
50	COMPARE	200, 201	Compare v and w and set condition codes
51	JUMPGE	54	If v ≥ w go to address 54
52	MOVE	204, 202	Otherwise give x the value of z
53	JUMP	55	Go to address 55
54	MOVE	203, 202	Give x the value of y
55			The next instruction begins here

d. while y < z do

```
    set y to value (y + w + z)
    set z to value (z + v)
end of loop
```

Memory Location	*Op Code*	*Address Field*	*Comment*
50	COMPARE	203, 204	Compare y and z and set condition codes
51	JUMPLT	53	If y < z go to address 53
52	JUMP	57	Otherwise, go to address 57
53	ADD	203, 201, 203	y now contains the value y + w
54	ADD	203, 204, 203	y now contains the value y + w + z
55	ADD	204, 200, 204	z now contains the value z + v
56	JUMP	50	Go to address 50
57			The next instruction begins here

22. a.

Memory Location	*Op Code*	*Address Field*	*Comment*
50	LOAD	300	Register R now contains the value of a
51	ADD	301	R now contains the sum a + b
52	ADD	401	R now contains the sum a + b – 1
53	STORE	300	Store the result into a

b.

Memory Location	*Op Code*	*Address Field*	*Comment*
50	COMPARE	300, 402	Compare a and 0 and set condition codes
51	JUMPLE	54	If a ≤ 0 go to address 54
52	LOAD	400	Otherwise, register R now contains the value 1
53	STORE	301	Store 1 into b
54			The next instruction begins here

23. The error is on the second line. The instruction ADD 19 does not add the integer 19 to the value of y but, instead, the *contents* of memory cell 19. To correct this error you would need to store the constant +19 into a memory cell, say location 600, and then write the instruction ADD 600.

24. a. MOVE X, Y

1. $IR_{addr1} \rightarrow MAR$	Send address of X, currently in IR_{addr1}, to the MAR
2. FETCH	Fetch the contents of cell X and put in the MDR
3. $IR_{addr2} \rightarrow MAR$	Send address of Y, currently in IR_{addr2}, to the MAR
4. STORE	Store the value in the MDR into memory cell Y

b. ADD X, Y

1.	$IR_{addr1} \rightarrow MAR$	Send address of X, currently in IR_{addr1}, to the MAR
2.	FETCH	Fetch the contents of cell X and put in the MDR
3.	$MDR \rightarrow ALU$	Send contents of MDR to the ALU
4.	$IR_{addr2} \rightarrow MAR$	Send address of Y, currently in IR_{addr2}, to the MAR
5.	FETCH	Fetch the contents of cell Y and put in the MDR
6.	$MDR \rightarrow ALU$	Send contents of MDR to the ALU
7.	ADD	Activate ALU and select output of add circuit
8.	$ALU \rightarrow MDR$	Copy output of the ALU to the MDR
9.	STORE	Store the value in the MDR into memory cell Y, whose address is still in the MAR

25. ADD X, v, Y

1.	$IR_{addr1} \rightarrow MAR$	Send address of X, currently in IR_{addr1}, to the MAR
2.	FETCH	Fetch the contents of cell X and put in the MDR
3.	$MDR \rightarrow ALU$	Send contents of MDR to the ALU
4.	$IR_{addr2} \rightarrow ALU$	Send the second address field directly into the ALU without fetching since it is a constant, not an address
5.	ADD	Activate ALU and select output of add circuit; the ALU now holds CON(X)+v
6.	$ALU \rightarrow MDR$	Copy ALU to the MDR. MDR now holds CON(X)+v
7.	$IR_{addr3} \rightarrow MAR$	Send address of Y, currently in IR_{addr3}, to the MAR
8.	STORE	Store the value CON(X)+v in the MDR into memory cell Y which is in the MAR

26. What makes the SETI project so well suited for grid computing is that the data can be easily divided into a large number of independent subsets, each of which can be analyzed in its entirety without having to communicate with other systems. When Arecibo sends out a chunk of radio telescope data, that data can be analyzed by a single computer to determine whether it does or does not contain the special pattern for which it is searching. When it is done, it only needs to communicate a single word message: Yes (if it found something) or No (it did not). That independence makes it an ideal research project to divide up into the hundreds of thousands of pieces of data that will be sent out to the hundreds of thousands of computers that are part of BOINC.

Challenge Work

1. With 100 processors, one can perform up to 100 addition operations at each step. One possible algorithm will use up to 50 processors simultaneously.

In the while loop:

```
While i < 101 do the following
```

instead of

$Sum = Sum + a_i$,

one could perform 50 operations simultaneously

($sum_1 = a_1 + a_2$
$sum_2 = a_3 + a_4$
$sum_3 = a_5 + a_6$
.
.
.
$sum_{50} = a_{99} + a_{100}$)

resulting in 50 new sums.

On iteration 2, one could perform 25 operations simultaneously.

($secondsum_1=sum_1+sum_2$
$secondsum_2=sum_3+sum_4$
.
.
.
$secondsum_{25}=sum_{49}+sum_{50}$)

resulting in 25 new sums.

Iteration 3 adds 12 pairs giving 12 new sums plus 1 remaining from the previous level.
Iteration 4 adds 6 pairs giving 6 new sums plus 1 remaining from the previous level.
Iteration 5 adds 3 pairs giving 3 new sums plus 1 remaining from the previous level.
Iteration 6 adds 2 pairs giving 2 new sums.
Iteration 7 adds 1 pair, giving the final result.

Instead of 100 iterations, 7 iterations will be needed, which means the algorithm with the parallel processing is over 14 times faster. This is logarithmic time instead of the linear time required to do the job sequentially.

No more than 50 processors were needed and any more than 50 processors would be too many for this specific problem.

2. Students should choose a processor and use the Internet to find information regarding the specific architecture chosen, although it may be difficult to find some of these details. Stress writing skills here so that you get a report, not merely a list of attributes.

Chapter 6: An Introduction to System Software and Virtual Machines

1. Answers may vary. The user interface of any device is that part most visible to the user and with which the user directly interacts. Examples are the labeled buttons of a DVD's remote control, the LED of a copier, and preset defrost/cook buttons on a microwave.

2. Answers may vary. Cases where one might like to see the underlying hardware of the computer system are when one wants to see and manipulate the details of machine operation, typically when writing system software such as an assembler or a linker or an operating system. Other instances may occur when interfacing with a device such as a mouse or a graphics tablet. There are programming languages (one example is C–see Chapter 10) that allow you to do machine-level manipulations, and one typically uses such a language for writing system software.

3. The components here are the English letter, the Spanish letter, and the translator. The English letter corresponds to the source program, the Spanish letter corresponds to the object program, and the translator corresponds to the assembler.

4.

a. LOAD 60	R: 472	60: 472	61: -1
b. STORE 60	R: 13	60: 13	61: -1
c. ADD 60	R: 485	60: 472	61: -1
d. COMPARE 61	R: 13	60: 472	61: -1
e. IN 61	R: 13	60: 472	61: 50
f. OUT 61	R: 13	60: 472	61: -1

5. a. 6
b. the value in address 6
c. 6
d. the value in address 7

6. If we execute .DATA 16387 then the processor will fetch, decode, and attempt to execute the "instruction" 16387. The binary representation of 16387 is

0100 0000 0000 0011

The first 4 bits would be the op code for the INCREMENT operation, while the address field would be interpreted as the value 3. So, the contents of memory location 3 would be incremented by 1.

7. a. SUBTRACT 780
b. ADD 7

8. THREE: .DATA 2 is perfectly legal, but it doesn't make much sense. It assigns the label THREE to a memory cell that contains the value 2. An instruction such as ADD THREE would therefore add 2 to whatever is in register R. While the computer would have no problem, a reader of the program would probably be misled.

9. Assume that K, L, M, N have already been given values

a.
```
                LOAD       K
                ADD        THREE
                STORE      K
                 .
                 .
                 .
   THREE:       .DATA      3
    K:          .DATA      …
```

b.
```
                LOAD       L
                ADD        ONE
                SUBTRACT   M
                SUBTRACT   N
                STORE      K
                .
                .
                .
    ONE:        .DATA      1
    K:          .DATA      …
    L:          .DATA      …
    M:          .DATA      …
    N:          .DATA      …
```

c.
```
                LOAD       TEN
                COMPARE    K
                JUMPGT     OUTPUT
                JUMP       NEXT
    OUTPUT:     OUT        K
    NEXT:       next instruction goes here…
                .
                .
                .
    TEN:        .DATA      10
     K:         .DATA      …
```

d.
```
                LOAD       L
                COMPARE    K
                JUMPGT     GREATER
                OUT        L
                INCREMENT  L
                JUMP       NEXT
  GREATER:      OUT        K
                INCREMENT  K
  NEXT:         next instruction goes here…
                    .
                    .
```

```
                  .
K:                .DATA …
L:                .DATA …
```

e.
```
                LOAD        ONE
                STORE       K
                LOAD        HUNDRED
REPEAT:         OUT         K
                INCREMENT K
                COMPARE     K
                JUMPGT      NEXT
                JUMP        REPEAT
NEXT:           next instruction goes here…
                   .
                   .
                   .
ONE:            .DATA 1
HUNDRED:        .DATA 100
K:              .DATA …
```

10. The difference is that the left-hand set of instructions will only add two to the register copy of X, leaving the original value of X unchanged. The right-hand set of instructions adds two to the value in memory cell X.

11. CLEAR SUM is not necessary because SUM has been set to 0 with the .DATA pseudo operation. LOAD ZERO is necessary because the register must be reset at the start of each loop to test whether each new number is nonnegative.

12.
```
                .BEGIN
                CLEAR       SUM
                CLEAR       SUM2
                IN          N           --Read in next value
AGAIN:          LOAD        ZERO
                COMPARE     N
                JUMPLT      NEG
                JUMPEQ      DONE        --if N = 0, finish up
                LOAD        SUM         --N is a positive number
                ADD         N           --add N to positive sum
                STORE       SUM
                IN          N           --read next N and repeat
                JUMP        AGAIN
NEG:            LOAD        SUM2        --N is negative
                ADD         N           --add N to negative sum
                STORE       SUM2
                IN          N           --read next N and repeat
                JUMP        AGAIN
DONE:           OUT         SUM         --write positive sum
                OUT         SUM2        --write negative sum
                HALT
SUM:            .DATA       0
SUM2:           .DATA       0
N:              .DATA       0
ZERO:           .DATA       0
```

13.

```
                .BEGIN
                IN          MAX         --read in first value as MAX
                LOAD        MAX
                STORE       MIN         --store first value as MIN
AGAIN:          INCREMENT COUNTER       --going to read another value
                IN          N
                LOAD        N
                COMPARE     MAX
                JUMPLT      LARGER      --MAX < N, N should be new MAX
                COMPARE     MIN
                JUMPGT      SMALLER     --MIN > N, N should be new MIN
                JUMP        TEST        --N is neither new MAX nor new MIN
LARGER:         STORE       MAX         --store N in MAX
                JUMP        TEST
SMALLER:        STORE       MIN         --store N in MIN
TEST:           LOAD        COUNTER
                COMPARE     HUNDRED
                JUMPGT      AGAIN       --less than 100 values read
DONE:           OUT         MAX         --write out max and min
                OUT         MIN
                HALT
N:              .DATA       0
MAX:            .DATA       0
MIN:            .DATA       0
COUNTER:        .DATA       1
HUNDRED:        .DATA       100
```

14. a. The list of 16 op codes would be sequentially searched for each of the 100 instructions. In the worst case each search would require examining all 16 op codes, giving $100 \times 16 = 1600$ comparisons. Therefore the time taken would be 1,600 comparisons × 1/50,000 seconds/comparison = 0.032 seconds.

b. Using binary search on a list of 16 op codes (the list of op codes must be sorted) requires in the worst case $1 + \lfloor \lg 16 \rfloor = 5$ comparisons (see Exercise 31 of Chapter 3). Therefore the time taken would be 100 × 5 comparisons × 1/50,000 seconds/comparison = 0.01 seconds

Using binary search is slightly faster.

With 300 op codes, the results are:

a. $100 \times 300/50000 = 0.6$ seconds
b. $100\times (1 + \lfloor \lg 300 \rfloor)/50000 = 100\times 9/50000 = 0.018$ seconds. Binary search is more than 30 times faster.

With 50,000 instructions, the results are:

a. $50000 \times 16/50000 = 16$ seconds
b. $50000 \times (1 + \lfloor \lg 16 \rfloor) /50000 = 50000 \times 5/50000 = 5$ seconds. Binary search is more than 3 times faster.

15. This is how the symbol table will look:

Symbol	Address
AGAIN	2
ANS	7
X	8
ONE	9

16. This is how the symbol table will look:

Symbol	Address
AGAIN	2
NEG	10
SUM	12
N	13
ZERO	14

17. This pair of statements is illegal – as figure 6.11 shows, the assembler checks in pass 1 for using the same symbol name twice, which is illegal. It is illegal because each symbol can be associated with only a single numeric memory address.

18. Some drawbacks to using passwords to limit access to a computer system are the possibility of having the password forgotten, stolen, or guessed. Using authorization lists, an operating system can restrict access to certain folders or files by legitimately logged-on users, as described in this chapter. This is appropriate when users of different standing have access to the same machine (for example a supervisor might have access to more files than a clerk). For highly sensitive data, the operating system might require a physical identification test such as a fingerprint or retinal scan.

19. Authorization lists must be encrypted and protected from unauthorized change because they restrict unauthorized users from viewing information that is confidential and valuable. If the authorization list is modified, it could allow users to access data they should not see and modify

or even to delete it. Also, users could be denied access to files that they have a legitimate right to read and change.

20. If 3 programs, A, B, and C, are loaded into memory, then for the processor to be idle at any time, all three programs must be waiting for I/O operations. Each program spends 1/2 of its time waiting for an I/O operation, so at any moment of time, there is a $(1/2)\times(1/2) \times (1/2) = 1/8$ probability that all three programs are waiting. Therefore the processor is doing useful work $1 - (1/8) = 7/8$ of the time, which is 87.5%. With 4 programs in memory the processor is doing useful work $1 - (1/2)^4 = 1 - 1/16 = 15/16 = 93.75\%$. With 5 programs the processor is doing useful work $1 - (1/2)^5 = 1 - (1/32) = 31/32 = 96.875\%$, more than 95%.

21. If two friends call each other at the exact time, both will receive busy signals and will hang up. Both will redial 1 minute later and the same thing will happen again; in fact, this will go on indefinitely. This problem could be solved (or at least made highly unlikely) by waiting a random amount of time before redialing.

22. A batch operating system would be far too slow to handle such modern applications, which require real-time responses. Also, in an application such as an airline reservations system, many users need access to the software at the same time.

23. In a time-sharing operating system, if the time slice value is too large, a user may wait a long time before his or her program gets any processing time. If the time slice is too small, each user's turn comes around quickly but very little progress can be made during that turn because of the small time slice; in addition, the processor will spend a larger proportion of its time swapping between jobs.

24. When a user has his or her own machine (i.e., a personal computer, tablet, smartphone) there is no delay to wait for the processor to turn its attention to the user's program. The user is never competing with other users for processing time and computing is slowed only by the I/O operations.

25. Answers may vary. The user may need additional privileges to access a remote resource, and the response may be slower than for a local resource.

26. The operating system should handle these requests in the following order, with the most critical or time-sensitive requests being handled first:

1. Someone pulled the plug on the power supply, and the system will run out of power in 50 msec. (This is critical or the entire system will shut down.)
2. The program running on processor 2 is trying to perform an illegal operation code. (An illegal operation code could write over data or program instructions.)
3. The clock in the computer has just “ticked” and we need to update a seconds counter.
4. The disk has just read the character that passed under the read/write head, and it wants to store it in memory before the next one arrives. (Least critical - external data is accessed at a slow rate compared to processor operations and the character can be stored in a local buffer until other crises have passed and it can be written to memory.)

Challenge Work

1. This is indeed a challenging problem. Note that you do not want to repeatedly read a value and add it to a sum, similar to what is done in Figure 6.8. Instead you want to read all 50 values into memory and then access each one in turn. The assembly language program will need 50 memory cells to store the 50 values, but you do not want to make 50 .DATA pseudo-op statements nor have to write 50 input statements. The declaration

```
A:      .DATA 0
```

will reserve a memory location, let's say it is memory location 42, with the label A for the first of the 50 values to be stored. Make this pseudo-op the last program instruction so that subsequent memory locations are available to store the remaining 49 numeric values.

The first of the 50 values will be read into location A using an instruction such as

```
REPEAT:         IN      A
```

REPEAT is then a label attached to a particular memory address. The content of that memory address is the instruction

```
op code       address
  1101    0000 0010 1010
```

but this instruction can be treated as data. In particular, the statement

```
INCREMENT REPEAT
```

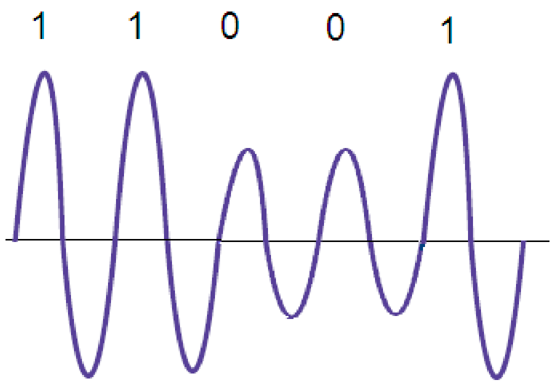
says to increment the contents of memory cell REPEAT, which would change the REPEAT statement to

```
op code       address
1101    0000 0010 1011
```

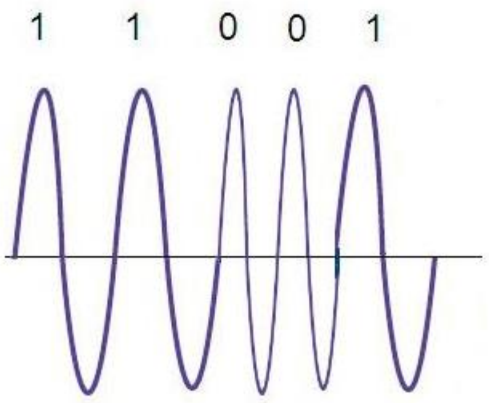
which means "input to 43, the next address after A". Hence a loop containing the INCREMENT REPEAT instruction can be used to read in the 50 values. A similar loop can be used to add each value in the list to the running sum.

Chapter 7: Computer Networks, the Internet, and the World Wide Web

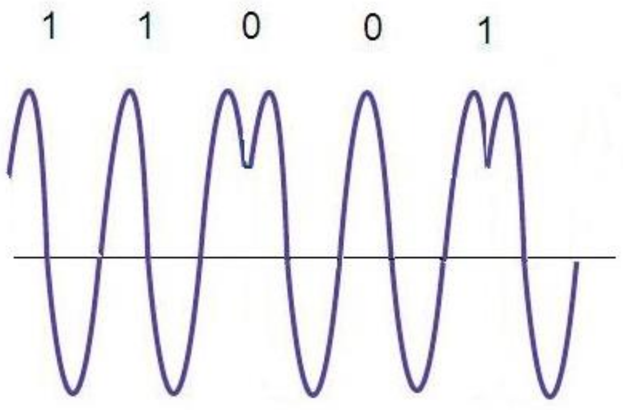
1. a. Here 1 is represented with a high amplitude (height of the wave) and 0 with a low amplitude.



b. Here 1 is represented with a low frequency (number of waves per second) and 0 with a high frequency.



2. The phase of a carrier wave is the left-right location of the wave. Phase modulation occurs when the carrier wave is shifted left from its standard position. A change in binary signal from 1 to 0 or 0 to 1 is detected by this change of "shift" in phase.



3. In an analog environment a signal can take on any value so it is theoretically impossible to determine exactly how much noise has been added to the carrier wave, and it is difficult to reconstruct its original uncorrupted shape. However, with a digital signal there is only a discrete set of allowable values so it may be possible to restore the corrupted signal to its original state. For example, if 0 volts represents logical zero while 10 volts is a logical 1, we could take a corrupted signal and say that all values from 0-5 volts are to be reset to 0 volts (logical 0) while all values 5.01 volts and above are to be reset to 10 volts (logical 1). While we might be wrong in our reconstruction, there is a good likelihood that we will be able to reconstruct the original signal exactly.

4. The total number of pixels is 1280 × 840 = 1,075,200 pixels
The total number of total bits to transfer is 1,075,200 × 8 = 8,601,600
a. A 56 Kbps modem
8,601,600 bits /56,000 bits per second = 153.6 seconds
b. A 1.5 Mbps DSL line
8,601,600 bits /1,500,000 bits per second = 5.7344 seconds
c. A 100 Mbps Ethernet link
8,601,600 bits/100,000,000 bits per second = 0.086016 seconds

5. 1440 × 900 = 1.296 million pixels × 16 bits/pixel = 20.736 million bits. In order to transmit this amount of information in a time of 0.01 seconds, we need a link speed X such that:

20.736×10^6 bits ÷ X bits/second = 0.01 seconds
X = 2.0736 gigabits/second

6. a. Recall from chapter 5 that a megabyte can store about 2.5 books. Therefore, to store one million books, we would need 1,000,000 books / 2.5 books/Mbyte = 400,000 Mbytes = 400 Gbytes = 400,000,000,000 bytes.
b. The number of bits to be transferred is 400,000,000,000 × 8 = 3.2×10^{12} bits = 3,200,000 Mbits. At the rate of 10 Mbps, the transfer time would be
3,200,000 Mbits / 10 Mbits/second = 320,000 seconds = 3.7 days
At the rate of 1 Gbps = 1000 Mbps, the transfer time would be

3,200,000 Mbits / 1000 Mbits/second = 3,200 seconds = 53 minutes

7. The address field is needed in an Ethernet LAN protocol so that the message may be directed to one particular machine. Each machine receives the broadcast message, checks the address field, and accepts the message only if the address field contains its own address. One may want to either omit the address field entirely or use some "special" address value in the address field in the special situation where a message is being sent to every computer on the network.

8. In a very heavily loaded network, the Ethernet protocol would perform poorly. Many computers will be waiting for line access and then attempting to send simultaneously, which will result in many collisions. Then they must wait random amounts of time before attempting to retransmit their messages. The network user will see a slow response time.

9. The advantage to creation of a centralized LAN in which one node would be in charge of all decisions about who can send a message and who must wait is that there would be no collisions. The single node will schedule all the nodes' message sending activities, and each node can do other useful work until its scheduled time to transmit occurs, rather than continuously testing for the opportunity to access the line.

10. While the scheme of waiting a random amount of time to retransmit after a collision may diminish the chances of future collisions, there is no fixed time limit within which a machine is guaranteed the ability to send. It is theoretically possible to have a sufficient number of collisions that any maximum time value T can be exceeded.

11. a. The smallest number of point-to-point communication links such that every node in the network is able to talk to every other node is N-1.

b. The type of interconnection structure that should be used if one is worried about having a disconnected network is one with redundant paths between any two nodes. In the extreme, the network could be completely connected, where each node is connected to every other node.

12. The main advantage is redundancy in cases of 1) failure or 2) heavy traffic. If there are two distinct paths from A to B then a failure of a line along one of those links may not necessarily make it impossible for these two nodes to communicate. Secondly, if there is a good deal of traffic between A and B we can split that traffic between the two (or more) different routes and not cause as much congestion as if we had routed all of the traffic on the same path.

13. The store-and-forward protocol would send a message from A to B and wait for the acknowledgement message. When it did not come back it would retransmit the message, over and over. As described in the chapter the protocol would result in a never ending cycle of sending messages and waiting for the ACK that never will come. A modification we could make to this protocol to handle this particular situation would be to set an upper bound on the number of attempts we make along any given path. After N transmissions and N failures to receive an acknowledge packet from B, A can then look for another path on which to retransmit. If there are no alternative paths then the message cannot be delivered and A must give up.

14. A modification to the protocol that would make it more efficient and not cause the sender to have to stop each time it sends a message is to have the sender continually send packets and store local copies. When an ACK is received for a packet, the sender checks the sequence number and discards the local copy of that packet. Packets for which an ACK is not received can in time be retransmitted. In this protocol the receiver could receive packets out-of-order, but can use the sequence number to reassemble the packets in the correct order.

15. Using an ARQ algorithm, the sender would need to send 100 copies of the message with a separate address for each copy. Each copy would be forwarded through the network to its appropriate destination. This is much less efficient than in an Ethernet network which is basically designed as a broadcast medium.

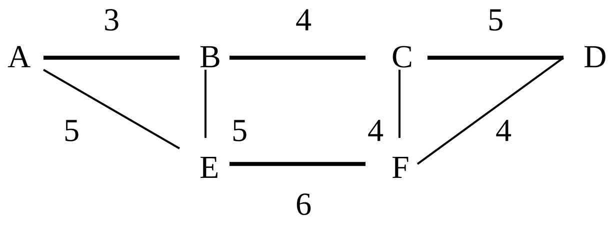
16. a. The number of simple paths from node A to G is 7.

ABCG
ABECG
ABEDCG
AFG
ADCG
ADEBCG
ADECG

b. There are two shortest paths from A to G, ADECG and ABECG, both of which have a cost of 10.

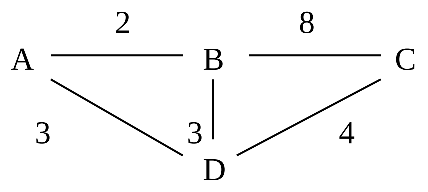
c. If node E fails, then both of the shortest paths fail, as do several of the other paths. The only remaining paths are ABCG, ADCG, and AFG, all of which have a cost of 11.

17. There are times when this heuristic will work and get the message to its intended destination in a reasonable time. For example, assume we want to route a message from A to D using the following network:



The heuristic starts out by routing the message from A to B, the lowest cost link of the two that go out from A. Then it selects BC because that is the lowest cost line of those that do not immediately return to A. Next it selects CF, since it is the lowest cost line that does not go directly back to B. Finally, it selects the line FD, and it has arrived at its destination via the route ABCFD. The total cost of this route is 3+4+4+4 = 15 units. This is not the lowest cost path, which would be ABCD at a cost of 3+4+5 = 12 units. But it did do a reasonably good job.

However, the question asks whether it *always* delivers the message to its destination, and the answer to that is no. For example, what if we want to deliver a message from A to C in the following configuration:



As it currently stands, the algorithm will route the message from A to B, then to D, then back to A, making an infinite number of repetitions of the cycle ABDABDABDABD…

18. A gateway's responsibilities include making the internetwork connections and providing routing between different WANs. More specifically, gateways must convert between different addressing schemes, message formats, and send/receive protocols. The advent of the standard TCP/IP protocol has made this task easier.

19. The data-link layer can only handle packets between nodes that are directly connected. It is the job of the network layer protocols to deliver a message from where it was created to its ultimate destination. The network layer might misinterpret an address and route the message to C instead of D. It might fail to route around a broken link between B and D or a failed node at B.

20. The advantages are:

a. You can send each piece via a different route to help balance traffic
b. If there is an error in a single packet it only destroys that packet, not the entire message
c. You can better service your traffic because you can send one packet from program A, then a packet from program B, etc. This gives each program a little piece of service rather than making it wait for an entire message to be sent.

21. Research on the Web required.

22.

a. When you store data locally it is available to you even when network access to the outside world is temporarily down. However, when you store data in the cloud you must have Internet access to get to your critically important data. Thus, when using the cloud you become totally dependent on a reliable network connection and highly vulnerable to network outages, which could bring your business to a halt.

b. When data is stored locally you are free to choose the level of security you wish to provide. For example, do you want to encrypt your data and, if so, using what strength encryption algorithm? What kind of authentication procedure do you wish to implement? These are all decisions that you can make. However, when data is stored in the cloud, you must accept the security procedures and protocols provided for you by your cloud provider. Hopefully, they will be acceptable and will meet your security needs, but if not there is nothing you can do to change them.

c. Your cloud provider is the one who decides exactly which data formats will be accepted for storage on its servers. For example, if you need to back up photographic information will your cloud provider accept JPG, GIF, and TIFF images? What about PNG or BMP? These decisions are out of your control. This is especially important when you wish to store "legacy" data–information represented in an older format that is no longer used. Your cloud provider may not be willing to accept data in this older format, requiring you to either store it locally or convert it to a different representation.

d. When data storage is handled locally, there is always a person on site you can contact to answer your questions about how to access a piece of information or fix a problem. When the data is moved off site, so is the customer support and help staff. Now to answer a problem, you must call your cloud service provider and hope that they have sufficiently knowledgeable personnel available to answer your question. Otherwise you may be sitting on hold for hours or talking to someone overseas who has no idea how to solve your problem.

Challenge Work

Research on the Web required. Most documentation available online is probably too detailed and technical for most students. You might look for resources aimed at the novice user before assigning this problem.

Chapter 8: Information Security

1. Scenario (a) tells the person attempting to log on the name of the computing system and that there is no valid user name “jones”. Scenario (b) tells the person attempting to logon the name of the computing system but does not reveal which part of the “smith/password” combination is incorrect. Scenario (c) gives no information, not even the name of the computing system, until the correct userID/password is given.

2.

a. Step 1: fido → 6 9 4 15
   Step 2: 6 + 9 + 4 + 15 = 34
   Step 3: 34 → 6 (34 = 4 × 7 with 6 left over)
   Step 4: (6 + 1) × 9 = 63
   Step 5: 63 → 36 → cf

b. Step 1: blank → 2 12 1 14 11
   Step 2: 2 + 12 + 1 + 14 + 11 = 40
   Step 3: 40 → 5 (40 = 5 × 7 with 5 left over)
   Step 4: (5 + 1) × 9 = 54
   Step 5: 54 → 45 → de

c. Step 1: ti34pper → 20 9 3 4 16 16 5 18
   Step 2: 20 + 9 + 3 + 4 + 16 + 16 + 5 + 18 = 91
   Step 3: 91 → 0 (91 = 13 × 7 with 0 left over)
   Step 4: (0 + 1) × 9 = 9
   Step 5: 9 → 9 → i

3. a. u s e r → 21 19 5 18 → 43 39 11 37 → 43039011037

b. It is a bad hashing algorithm because it is completely reversible. Given the above string, it can be separated (because of the 0s) into four individual numbers, which reverses Step 3:

43 39 11 37

For each of these numbers, Step 2 can be reversed by subtracting 1, then dividing by 2:

21 19 5 18

Then reverse Step 1 by translating these numbers back into letters: u s e r

Therefore the original password can be recovered from the encrypted password.

4. a. There are 10 possibilities for each digit, so there are 10 × 10 × 10 × 10 = 10,000 possible passcodes.

b. 10,000 seconds, or about 2.78 hours

5. The number n of possible characters is 26 + 26 + 10 + 3 = 65, and the maximum length k of the password is 10. The total number of passwords, using the formula given in Section 8.2.1, is $\frac{65(65^{10}-1)}{65-1}$ = 1.36731E+18. At the rate of 10,000,000 tries per second, it would take over 4,000 years to generate and test all potential passwords.

6. 0.132353 seconds

7. This can be seen by constructing a table showing the number of infected machines at the beginning and end of each hour.

Hour	Beginning of hour	End of hour
1	1	(1 × 50) × 10% = 5 = 5^1
2	5	(5 × 50) × 10% = 25 = 5^2
3	25	(25 × 50) × 10% = 125 = 5^3
4	125	(125 × 50) × 10% = 625 = 5^4
5	625	(625 × 50) × 10% = 3125 = 5^5

At the end of the fifth hour, 3,125 machines have been infected. At the end of n hours, 5^n machines will be infected.

8. Here is the message:

We here at Marriott appreciate your loyalty as a customer. We want to make things more easy for you when you travel, so we have partnered with Hilton and Ritz-Carlton to create a unified rewards program. Now when you stay at any of these fine brand hotels, you will earn award points that can apply to a future stay at any of the three hotels. For you we will quick set this up, just click on the link below to get started:

http://www.Mariott.com

Clues:

1. You should always be suspicious when a company asks you to go to a website and enter personal information.
2. Grammar error #1: " make things more easy for you"
3. Grammar error #2: "we will quick set this up"
4. In the URL: Marriott is misspelled as Mariott

9. Information is available on the Web.

10. a. Existing probability of failure per year is 2 events/4 years = 0.5 per year.

Existing risk exposure = ($600 + $12,000) * 0.5 = $6300.

Risk exposure with backup server = (\$600 + \$12,000) *0.2 = \$2520

The difference is \$6300 - \$2520 = \$3780, more than the cost of the new server. This would be a cost-effective measure.

b. With the new probability, the risk exposure with backup server =

(\$600 + \$12,000) *0.3 = \$3780.

The difference is \$6300 - \$3780 = \$2520, less than the cost of the new server. This would not be a cost-effective measure.

11. MOVE OUT AT DAWN

12. The s value is 12 and the decoded message is
ALL GAUL IS DIVIDED

13. a. $\begin{bmatrix} 5 & 24 \\ 19 & 3 \end{bmatrix} \times \begin{bmatrix} 3 & 2 \\ 7 & 5 \end{bmatrix} = \begin{bmatrix} 183 & 130 \\ 78 & 53 \end{bmatrix} \to \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ (modulo 26)

b. ATTACK

14. 100111 ⊕ 110101 = 010010

15. a. 1.25hours

b. 0.77 seconds

16. 1. n = p*q = 7*11 = 77

2. m = (p – 1)*(q – 1) = 6*10 = 60

3. Choose e = 13 (e = 13 and m = 60 have no common factors)

4. Then d = 37 because e*d = 37*13 = 481 = 8*60 + 1, so when we compute e*d using wrap-around arithmetic with respect to 60, we get 1.

Another possibility is e = 11, d = 11

17. a. We must have d between 0 and m = 8; d = 3 because e*d = 11*3 = 33 = 4*8 + 1.

b. The code for 3 is 3^{11} using wrap-around arithmetic with respect to 15. 3^{11} = 177147 = 11809*15 + 12 $\to$ 12.

c. To decode, compute $(12)^3$ using wrap-around arithmetic with respect to 15. 12^3 = 1728 = 115*15 + 3 $\to$ 3.

18. 256 qubits could simultaneously represent all 256 possible keys.

Challenge Work

1. The result when a message is hashed is also called a message digest, which may give students another term to search for.

2. It is doubtful that a student can answer all of these questions about any virus, but some of this information should be available.

3. The plaintext is ONEIFBYLAND.

Chapter 9: Introduction to High-level Language Programming

Each online language module has its own set of exercises. Chapter 9 in the text itself contains only Challenge Exercises.

Challenge Work

1. Students have the code in various languages for the array declaration and the input process from Section 9.3.2. After that it should not be too difficult to implement the pseudocode algorithm of Figure 3.1 in the chosen language.

2. This is a much more challenging problem. A C++ solution appears below. Other versions will be more or less similar.

```
//program to illustrate Caesar cypher
#include <iostream>
using namespace std;

void Encode(char message[], int max, int s)
{
   int i;           //counter for array position
   i = 0;
   while (i <= max)
   {
        message[i] = message[i] + s;
        if (message[i] > 'Z')
        {
              message[i] = message[i] - 26;
        }
        i = i + 1;
   }
}

void Decode(char message[], int max, int s)
{
   int i;           //counter for array position
   i = 0;
   while (i <= max)
   {
        message[i] = message[i] - s;
        if (message[i] < 'A')
        {
              message[i] = message[i] + 26;
        }
        i = i + 1;
   }
}
```

```
void WriteMessage(char message[], int max)
{
  for (int i = 0; i <= max; i++)
  {
     cout << message[i];
  }
  cout << endl;
}

void main()
{
  char message[10];     //array to store message word
  char next;            //next character read in
  int i;                //counter for array position
  int max;              //maximum array element used
  int s;                //shift amount

  cout << "Enter your message.  Use all uppercase " << endl;
  cout << "letters, maximum 10; end with %" << endl;

  i = 0;
  cin >> next;
  while (next != '%')
  {
     message[i] = next;
     cin >> next;
     i = i + 1;
  }
  if (i > 10)           //message was too long
  {
     cout << "Message too long, run the program again" << endl;
  }
  else
  {
     max = i - 1;

     cout << "The original message is " << endl;
     WriteMessage(message, max);

     cout << endl;
     cout << "What is the shift amount for the cypher?" << endl;
     cin >> s;
     cout << endl;

     Encode(message, max, s);
     cout << "The encoded message is " << endl;
     WriteMessage(message, max);
     cout << endl;

     Decode(message, max, s);
     cout << "The decoded message is " << endl;
     WriteMessage(message, max);
     cout << endl;
  }
}
```

3. A major task is collecting the data from the user and filling the 2-D array. Each language has its own way to declare a 2-D array. For Floyd's algorithm itself, students should be able to translate the given pseudocode to their choice of language. The length of the shortest path from 2 to 4 is 4, using the path 2 to 3 to 4.

4. and **5**. The details of file I/O are language-specific, and students will need some help here. After that, they should be able to adapt their solution to Exercise 1 fairly easily. Code for Exercise 5 will be a bit more challenging.

6. Any links to get students started on any of these topics would be valuable.

Chapter 10: The Tower of Babel

1. 10 (1 + 2 + 3 + 4)

2. 8

3. $3^4 = 81$

4. 301

5. The integer 10 has been stored in memory address 500.

6.
```
int* intPointer;
intPointer = (int*) 1000;
*intPointer = SAM;
```

7. The teller object can allow a customer to deposit money into an account, withdraw money from an account, and can tell a customer the current balance in the account.

8. After a println statement, the screen cursor goes to the next line. After a print statement, the screen cursor stays on the same line so the user input appears right after the colon.

9. The output is

```
second is bigger
first is bigger
```

because the else clause consists of just one line.

10. a. (3 <= 3) && (7 > 5) This is TRUE (both parts are true)
b. (3 < 3) || (7 > 5) This is TRUE (the second part is true)
c. (4 < 1) && (3 > 2) This is FALSE (the first part is false)

11. a. To compute trajectories for a satellite launcher, Fortran would work the best because Fortran is best suited to programs that require heavy mathematical computation.

b. This application requires low-level access to the details of the communications between the input device and the computer. C was designed to give this kind of low-level access to the programmer.

c. To process the day's transactions at an ATM, COBOL would work best since COBOL was designed to handle data manipulation and report generation tasks.

12. It results in the names of all vendors from Dallas with whom the store has done less than $10,000 worth of business in the past quarter.

13.
```
SELECT CITY
FROM VENDOR
WHERE PURCHASE > 10000
```

14. The sentence "The red dog chased the brown cow across the green field." would appear with the word "red" in red font, the word "brown" in brown font, and the word "green" in green font.

16. 21

17.
```
(define (toofer input-list)
   (list (car (cdr input-list)) (car input-list)))
```

[Then (toofer (list 3 4 5 6)) results in (4 3)]

18. The output is 3. This function returns the size of the list.

19.
```
(define (cube x)
    (* (* x x) x))
(define (double x)
    (* 2 x))
(define (five x)
    (* 5 x))
(define (poly x)
    (+ (- (double (cube x)) (five x)) 1))
```

[Then (poly 2) results in 7]

20. The result of the function is 4*3*2*1 = 24. This function returns the factorial of the number input.

21.
```
a. earlier(lewisandclark, civilwar).
       true   b. earlier(worldwarII, firstmoonlanding).
     true
c. earlier( X, worldwarII).
     X = gettysburgaddress
      X = civilwar
      X = lewisandclark
```

22.
```
                eli               mary
                 \________________/
                         |
                        bill
        _________________|_____________
```

```
      /                    |                  \
    joe                  betty              sarah
```

23. fatherof(X,Y) :- male(X), parentof(X,Y).

24. daughterof(X,Y) :- female(X), parentof(Y,X).

25. a. ancestorof(X,Y) :- parentof(X,Y).
ancestorof(X,Y) :- parentof(X,Z), ancestorof(Z,Y).

b. ancestorof(X, sarah).
X = bill X = eli
X = Mary

26. Students can copy and paste both their programs and their results for each query into a Word or text document for grading purposes.

27. Assume that all units are μsec. Using the sequential algorithm, 7 comparisons will take place. Therefore it will take the sequential algorithm 7 μsec to complete the Find Largest algorithm.

Using the parallel algorithm, it will take

.003*8	top level split
.003*4	2nd level split
.003*2	3rd level split
1	4th level comparison
.001	pass up
1	3rd level comparison
.001	pass up
1	2nd level comparison
.001	pass up

Total = 3.045 μsec

Challenge Work

1. Students should be provided sufficient documentation about Visual Basic.NET to solve this problem. If their only programming experience has been the console programming of the online modules, they will enjoy dragging the user interface objects onto the form and programming a GUI.

2. This would be a good chance to remind students that they can't steal images from the web unless they are specifically available for use without fee or permission.

3. a. 1, 1, 2, 3, 5, 8

b. 1, 2, 3, 5, 8

c. With //, the same as C or C++

d. The Go compiler automatically inserts semicolons before compilation.

e. The statement

```
a, b := 0, 1
```

assigns the value 0 to a and the value 1 to b.

The statement

```
a, b = b, a+b
```

computes the value of each expression on the right of the equals sign (i.e., the existing value of b, and the value of a + b using the existing values of a and b), then assigns those two values to a and b, respectively. For example, if a has the value 2 and b has the value 3, the expression b has the value 3 and the expression a + b has the value 2 + 3 = 5. Then the computed b value (3) is assigned to a and the computed a+b value (5) is assigned to b.

4. The Prolog simulator at *http://swish.swi-prolog.org* returns

```
precedes(fdr, kennedy).
true
```

with an indication that there are more answers, but when you ask to see more answers, the system quickly aborts with the message "out of local stack".

The first part of the "precedes" definition means that the system will check whether before(fdr, kennedy) is a fact in the Prolog database, and it is indeed a fact, so the answer returned from this check is "true", as shown above. But Prolog will go on to check the second definition of "precedes" with the result that in this process, the simulator's processor runs out of space.

In the original version, when trying to match the rule

precedes(X,Y) :- before(X,Z), precedes(Z,Y).

if the second (recursive clause) is ever tested, it checks a "before" relation ahead of trying another recursive call and there is a finite number of "before" facts to check, so this process will eventually terminate. In the new version

precedes(X,Y) :- precedes(Z,Y), before(X,Z).

the recursive part of the rule is checked first. So, where X is fdr and Y is kennedy, trying to match Z with jefferson, for example, would mean checking

precedes(jefferson, kennedy), before(fdr,jefferson)

Checking the simple version of precedes(jefferson, kennedy), that is, before(jefferson,kennedy), fails so the next step is the recursive version precedes(Z,kennedy) and before(jefferson,Z), and the recursive part of the rule is checked first. So, trying to match Z with jefferson, for example, would mean checking

precedes(jefferson,kennedy) and before(jefferson,Z).

And we are now back where we were earlier. The program is in an infinite loop and runs out of space to store all the intermediate results.

Chapter 11: Compilers and Language Translation

1. a. ‘if’, ‘(‘, ‘a’, ‘==’, ‘b1’, ‘)’, ‘a’, ‘=’, ‘x’, ‘+’, ‘y’, ‘;’
b. ‘delta’, ‘=’, ‘epsilon’, ‘+’, ‘1.23’, ‘-‘, ‘sqrt’, ‘(‘, ‘zz’, ‘)’, ‘;’
c. ‘print’, ’(‘,’Q’,’)’,’;’

2. Because an underscore character can appear in a variable name and there are no spaces in the string, the scanner will classify the string as a single five-character token.

3. Because both { } and (* *) can be used to enclose comments, it is likely that the scanner would group both { and (* into a "begin comment" classification, and both } and *) into an "end comment" classification. However, that would allow a comment to begin with, say, a { but end with a *). To insure that you use matching symbols, it might instead assign each of the four symbols its own classification number.

4. a. limit = begin + end

Token	*Classification*
limit	1
=	3
begin	1
+	4
end	1

b. a = b – 1;

Token	*Classification*
a	1
=	3
b	1
-	5
1	2
;	6

c. if (c == 50) x = 1; else y = x + 44;

Token	*Classification*
if	8
(	10
c	1
==	7
50	2
)	11
x	1
=	3
1	2
;	6
else	9
y	1
=	3
x	1
+	4

44	2
;	6

d. thenelse == error -

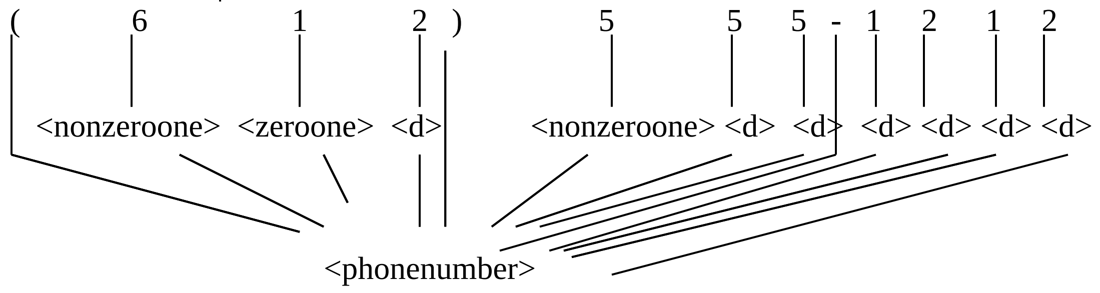
Token	*Classification*
thenelse	1
==	7
error	1
-	5

5. a. <number> ::= + <nzdigit> <digit> | <nzdigit> <digit>
<digit> ::= 0|<nzdigit>
<nzdigit> ::= 1|2|3|4|5|6|7|8|9

b.
```
+          9               0
|__________|_______________|
|          |               |
|     <nzdigit>         <digit>
 \ ________|______________/
           |
       <number>
```

6. a. <phonenumber> ::= (<d> <d> <d>) <d> <d> <d> - <d> <d> <d> <d>
| <d> <d> <d> - <d> <d> <d> <d>
<d> ::= 1|2|3|4|5|6|7|8|9|0

b. <phonenumber> ::= (<nonzeroone> <zeroone> <d>) <nonzeroone> <d> <d>
- <d> <d> <d> <d> | <nonzeroone> <d> <d> - <d> <d> <d> <d>
<d> ::= <zeroone>|<nonzeroone>
<nonzeroone> ::= 2|3|4|5|6|7|8|9
<zeroone> ::= 1|0

c.

(6 1 2) 5 5 5 - 1 2 1 2
<nonzeroone> <zeroone> <d> <nonzeroone> <d> <d> <d> <d> <d> <d>
<phonenumber>

7. a. <identifier> ::= <letter> <identifierbody> | <letter>
<identifierbody> ::= <symbol> <identifierbody> | <symbol>
<symbol>::=<letter>|<number>
<letter> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B…|Z
<number> ::= 0|1|2|3|4|5|6|7|8|9

```
b.  A              B           5        C           8
    |              |           |         \           \
  <letter>    <letter>   <number>  <letter>   <number>
    |              |           |           |           |
     \        <symbol>   <symbol>  <symbol>  <symbol>
       \           |           |           |           |
         \          |           |           |      <identifierbody>
          \         |           |           |           /
           \        |           |      <identifierbody>
            |       |           |           /
            |       |           |          /
            |       |     <identifierbody>
            |       |           /
            |       |          /
            |    <identifierbody>
            |        /
            |       /
         <identifier>
```

8. a.   <money> ::= $ <number> | $ <number> . <number> | $ <number> CR
                        | $ <number> . <number> CR
        <number> :: = <digit> | <digit> <number>
        <digit> ::= 0|1|2|3|4|5|6|7|8|9

b.   <money> ::= $ <number> | $ <number> . <digit> <digit> | $ <number> CR
                        | $ <number> . <digit> <digit> CR
        <number> :: = <digit> | <digit> <number>
        <digit> ::= 0|1|2|3|4|5|6|7|8|9

```
c.   $        1          9          .          9        5         CR
     |        |          |          |          |        |           |
     |        |          |          |          |         \          |
     |    <digit>   <digit>        |      <digit>  <digit>        |
     |        |          |          |          |           |        |
     |        |      <number>     |          |      <number>   |
     |        |       /             |          |       /            |
     |    <number>                 |      <number>            /
      \        \                    |         /                /
        \_______\___________|______/____________/
                                    |
                                <money>
```

9. The language consists of A and A,A

10. This language consists of A and A,A and AA,A and AAA,A and so forth. The
                <next> ::= <letter><next>
    rule allows arbitrarily long strings of A 's before the comma and the final A.

11. a. <Boolean> ::= <var> AND <var> | <var> OR <var>
<var> ::= w|x|y|z

b. <Boolean> ::= <expr> AND <expr> | <expr> OR <expr>
<expr> ::= <var> | (<var> == <var>) | (<var> < <var>) | (<var> > <var>)
<var> ::= w|x|y|z

c. <Booleanexpr> ::= <Boolean> | <Boolean> AND <Booleanexpr>
| <Boolean> OR <Booleanexpr>
<Boolean> ::= <expr> AND <expr> | <expr> OR <expr>
<expr> ::= <var> | (<var> == <var>) | (<var> < <var>) | (<var> > <var>)
<var> ::= w|x|y|z

```
12.  y    =     x         +          y         +         y       +         z
    |     |     |         \          |         |        |         |        |
    |      \     \          \        |        |         |        |        |
    |       |     \         |       |         |         /        |        |
<variable>   \  <variable>  |   <variable>    |  <variable>      |    <variable>
     \      /      |        |       |        |       |          /         |
     |      |      |        \       |        |       |         /        /
     |      | <expression>    \     |        |       |        /        /
     |      |       \         |     |       |      /         /        /
     |      |         \       |     /      /      /         /       /
     |      |           <expression>      /    /      ___  /      /
     |      |                 |       _ /    /        /         /
      \      \                |       /     /        /         /
       \      \               <expression>       __/          /
         \      \                  \            /            /
            \     \                <expression>  -----------/
             \     \                 /
               <assignment statement>
```

This parse tree is unique. The operations must be done from left to right.

13. This is the set of all strings of 1 or more binary digits. This is the set of all possible binary numbers.

14. <string> ::= Λ | <letter> <string>
<letter> ::= a | b | c

15. The two different interpretations of this ambiguous sentence are that the shirt was too large, or that the store was too large.

16. <complexinput> ::= input (<vars>)
<vars> ::= <var> | <var>, <vars>
<var> ::= <letter> | <letter> <char>
<char> ::= <letter> | <digit> | <letter> <char> | <digit> <char>
<letter> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|A|B|…|Z
<digit> ::= 0|1|2|3|4|5|6|7|8|9

17. Other information one might want to store in a semantic record, in addition to name and data type, would be whether the quantity is a constant, or, in the case of an array, the size of the array. Such information would come from the variable declaration**.**

18. This production does not generate any code because if it did, its effect would be to create storage for the variable x, that is, write a .DATA statement, but this was already done in the occurrence of x on the right side of the assignment operator.

19. The compiler would discover that it needs to do a data conversion when it tries to produce code to execute the assignment operation in y = x. At that point it would detect that the semantic record for x was type integer and the semantic record for y was type double. The binary representation of x as an integer would need to be converted to the binary representation of x as a real number, probably using some predefined assembly language code.

20. The concept of algebraic identities could be exploited during the code optimization phase of compilation by replacing an expression with an equivalent expression requiring fewer operations. For example, X + 0 could be replaced by X, reducing the number of operations from one to zero. This would be local optimization. Other examples would be

X * 1 = X	(reduce operations from one to zero)
X * Y + X * Z = X*(Y + Z)	(reduce operations from three to two)

21. The expression (a + b + c * 3) appears in both statements. The compiler could compute this value once, save it, and use it in both statements instead of having to compute it twice. These two statements might be separated by many other statements, so the compiler has to take a larger view of the code. This would be a global optimization.

22. If all mathematical operations take 5 nsec, then the optimization done in problem 21, which saves 3 operations, saves 15 nsec. This is not much time at all. Again, the choice of a more efficient algorithm, if available, will far outstrip the small improvements achieved by compiler optimization.

23. The compiler first checks the type fields to be sure that the two variables being compared are the same type. Then it creates a semantic record linked to <Boolean expression> that is named *temp* and that has a Boolean data type. The machine language code generated could be:

```
LOAD       a
COMPARE    b
JUMPEQ     EQUAL
LOAD       FALSE
```

```
         JUMP     DONE
EQUAL:   LOAD     TRUE
DONE:    STORE    TEMP
         .
         .
         .
A:       .DATA    0
B:       .DATA    0
TRUE:    .DATA    1
FALSE:   .DATA    0
TEMP:    .DATA    0
```

Challenge Work

1 and 2. Try to provide students with leads to documentation of LEX and YACC that are written at a level for novice readers.

Chapter 12: Models of Computation

1. Answers may vary. Factors that might be included are the incubation period, population density, time until death or recovery, probability of infection if exposed.

2. This is a subjective question, but students should be able to back up their answers with reasonable arguments. For example: In the first case the manufacturer is likely to be found accountable because it did not take what are considered standard testing steps (prototype vehicles). In the second case, the manufacturer may not be found accountable because the prototype is presumed to be closer to the real thing than a simulation, so it could be argued that if the fault is not detected in the prototype, it would not have been detected in a simulation.

3. a. modeling long-term effects on the body of some drug
b. modeling nutritional content of a food product under development
c. modeling potential risks/hazards and their effect on policy holders

4. a. yes - it takes and stores input (alarm time setting), takes action according to instructions (alarm, music, snooze alarm, etc.), produces output
b. no - it is a physical device that does not follow instructions
c. no - same as above
d. yes (it is clear that it has all four properties)

5. The next configuration is

```
b   0   0   b
↑
2
```

6. No. This machine moves only to the right, so in the previous configuration it had to be looking at the position now occupied by the 0. Neither instruction writes a 0, therefore the 0 character must have been present in the previous configuration. However, had the machine been in state 1 looking at a 0, it would have halted because there is no appropriate instruction.

7. No. There are two different instructions that have the form (3,0,-,-,-).

8. b 1 0 0 0 1 b

9. b 1 1 1 b

10. The Turing machine cycles forever back and forth over the two leftmost nonblank cells, alternating between states 1 and 2.

11. The Turing machine never halts, but keeps writing

```
    1 0 0 1
  1 0 0 0 1
1 0 0 0 0 1
```

etc. on the tape.

12. It passes over the original binary string, leaving it unchanged, and then moves forever right, changing blanks to 1s.

13.	(1,1,0,2,L)	change leftmost 1 to 0 and halt

14. No - this Turing machine only works on this particular input string, or a shorter string; it will not work on a longer string.

15. a.	(1,1,1,1,R)	pass right over all the 1s
	(1,b,b,2,L)	find the right end of the string, move left
	(2,1,0,2,R)	change rightmost 1 to 0

b. The solution to part (a) will work on any unary string

16.	(1,1,b,1,L)	remove a single 1 from the string (n > 0, so there are at least two 1s in the string to begin with)
17.	(1,1,b,2,R)	remove a single 1 from the string
	(2,b,0,1,R)	if n = 0, then tape contained a single 1, write a 0
18.	(1,1,1,1,R)	pass over any 1s
	(1,0,1,1,R)	change any 0s to 1s
19.	(1,1,1,2,R)	leave 1 alone, go to state 2
	(2,1,0,1,R)	change 1 to 0, go to state 1
20.	(1,1,1,2,R)	Even parity state reading 1, change state.
	(1,0,0,1,R)	Even parity state reading 0, don’t change state.
	(2,1,1,1,R)	Odd parity state reading 1, change state.
	(2,0,0,2,R)	Odd parity state reading 0, don’t change state.
	(1,b,0,3,R)	End of string in even parity state, write 0 and go to state 3.
	(2,b,1,3,R)	End of string in odd parity state, write 1 and go to state 3.
21.	(1,1,1,2,L)	Pass to the left over start of string
	(2,b,1,3,L)	Add the first 1 at the lefthand end of the string
	(3,b,1,4,L)	Add the second 1 at the lefthand end of the string
	(4,b,1,5,L)	Add the third 1 at the lefthand end of the string
22.	(1,1,1,1,R)	in state 1, find the right end of the string
	(1,0,0,1,R)	
	(1,b,0,2,L)	attach a 0 on the right end, prepare to move left
	(2,1,1,2,L)	in state 2, find the left end of the string
	(2,0,0,2,L)	
	(2,b,0,1,R)	attach a 0 on the left end, prepare to move right

23.	(1,1,1,2,R)	read initial 1
	(2,b,1,5,R)	number is 0, write the second 1 and halt
	(2,1,1,3,R)	number was not 0, move right
	(3,b,1,5,R)	if number is 1, write the third 1 and halt
	(3,1,1,4,R)	if number > 1, leave third 1 and move right
	(4,1,b,4,R)	erase any additional 1s

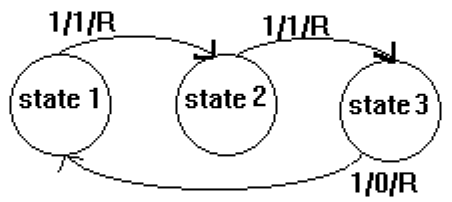
24.	(1,1,0,1,R)	change leftmost 1 to 0
	(1,1,1,1,R)	move right over any 1s
	(1,b,b,2,L)	
	(1,X,X,2,L)	right end of 1s found, move left
	(2,1,X,3,L)	mark rightmost 1 with X, move left
	(3,1,1,4,L)	more 1s remain in first half
	(4,1,1,4,L)	move to left end of 1s, start again
	(4,0,0,1,R)	
	(3,0,0,5,R)	no more 1s, midpoint found
	(5,X,1,5,R)	change markers in right half to 1s

25.	(1,1,X,2,R)	mark a 1 in first number, move right
	(2,1,1,2,R)	
	(2,b,b,3,R)	separating blank found
	(3,X,X,3,R)	move right over markers in second number
	(3,b,b,4,L)	first number larger, prepare to erase second number markers
	(4,X,b,4,L)	erase second number markers
	(4,b,b,5,L)	separating blank found
	(5,1,1,5,L)	
	(5,X,1,5,L)	restore first number
	(3,1,X,6,L)	second number at least as big as first, set marker, move left to start again
	(6,X,X,6,L)	move left over second number markers
	(6,b,b,7,L)	separating blank found, move left to first number
	(7,1,1,7,L)	move left over first number
	(7,X,X,1,R)	first number markers found, start over
	(1,b,b,8,L)	first number exhausted, so second number is larger

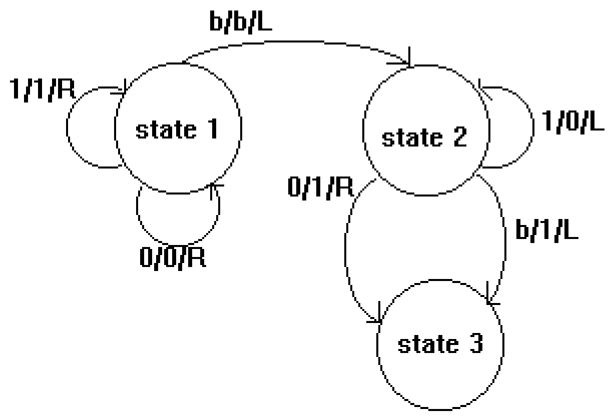
(8,X,X,8,L)	
(8,b,b,9,R)	find left end of string
(9,X,b,9,R)	erase first number
(9,b,b,10,R)	separating blank
(10,X,1,10,R)	restore second number

26. Yes, it simply erases the single 1 that represents the 0 and halts, leaving on the tape the representation of the second number.

27.



28.



29. The TM sweeps to the right, then to the left, changing 1s to 0s, then back to the right, changing 0s to 1s. Space efficiency is n since no new cells are used. Time efficiency is about 3n for the three sweeps of the input. The exact expression is 3n + 2, the 2 being the two turnarounds.

30. Multiply by using repeated additions. Use the first of the two numbers to count the number of additions. For each 1 in the first number (except the final one that is the extra representation digit), do a copy of the second number. Then add the copies together one by one.

31. a. yes

b.

(1,1,1,1,R)	move right to separating blank
(1,b,1,2,R)	fill blank with 1
(2,1,1,2,R)	move to far right end
(2,b,b,3,L)	

(3,1,b,4,L)	erase a 1
(4,1,b,5,L)	erase a 1

(1,1,b,2,R)	erase 1 on left end
(2,1,1,2,R)	move to blank
(2,b,1,3,R)	fill blank
(3,1,1,3,R)	move to far right end
(3,b,b,4,L)	
(4,1,b,5,L)	erase a 1

c. The one given in the chapter seems most efficient. Both of the other two require traveling all the way to the far right end.

d. algorithm from the chapter: n + 2 (n + 1 to traverse the representation of n plus 1 to fill in the blank)

algorithm 1: n + m + 3 (to traverse the tape) + 3 (to recognize the far right end and delete the two 1s)

algorithm 2: n + m + 3 (to traverse the tape) + 2 (to recognize the far right end and delete a 1)

The first is most efficient.

e. n + m + 3

32.

(1,1,b,2,R)	first character is a 1, blank it out and go to state 2
(1,0,b,3,R)	first character is a 0, go to state 3
(2,1,1,2,R)	
(2,0,0,2,R)	in state 2, find the right end of the string
(2,b,b,4,L)	
(3,1,1,3,R)	
(3,0,0,3,R)	in state 3, find the right end of the string
(3,b,b,6,L)	
(4,1,b,5,L)	last character is a 1 (matching first character), blank it out and go left in state 5
(4,0,0,4,R)	last character is a 0, not a palindrome, halt with nonblank tape
(6,0,b,5,L)	last character is a 0 (matching first character), blank it out and go left in state 5
(6,1,1,6,R)	last character is a 1, not a palindrome, halt with nonblank tape
(5,1,1,5,L)	

(5,0,0,5,L)	OK so far, go to left end and start again
(5,b,b,1,R)	

33. a. The language consists of strings of one or more 1's.

b.

(1,1,b,2,R)	One 1 has been read
(2,1,b,2,R)	Read any additional 1's. Any 0's on the tape cause the machine to halt with nonblank tape.

34. a. The language consists of all strings that start with zero or more pairs of 0's followed by a single 1.

b.

(1,0,b,2,R)	
(2,0,b,1,R)	Cycle back and forth between states 1 and 2 to read pairs of 0's
(1,1,b,3,R)	This should be the single 1 at the end, make sure there are no more 1's
(3,b,b,4,R)	no more 1's, machine halts

35. a. The language consists of all strings beginning with two 0's, followed by 1 or more 1's, followed by two 0's.

b.

(1,0,b,2,R)	Read the two 0's at the front
(2,0,b,3,R)	
(3,1,b,4,R)	Read the single required 1
(4,1,b,4,R)	Read any additional 1's
(4,0,b,5,R)	Read the two 0's at the back
(5,0,b,6,R)	

36. a. The language consists of strings beginning with some positive number of 0's followed by the same number of 1's

b.

(1,0,b,2,R)	Blank out first 0
(2,0,0,2,R)	Move right over remaining 0's and 1's
(2,1,1,3,R)	
(3,1,1,3,R)	
(3,b,b,4,L)	At far right end, move left
(4,1,b,5,L)	Blank out matching 1
(5,1,1,6,L)	Go to the left end and start again
(6,1,1,6,L)	
(6,0,0,7,L)	
(7,0,0,7,L)	
(7,b,b,1,R)	

37. Yes. The Church-Turing thesis says nothing about halting. But even the unsolvability of the halting problem says that there is no algorithm to test *all* program/input pairs, however there may well be a way to test a specific program/input pair.

38. It proves the existence of clearly-stated problems that have no algorithmic solution.

39. The first one.

40. The algorithm for the solution is to run the Turing machine on the tape and wait for 10 steps. If the machine has halted within that time, the answer is yes, otherwise the answer is no. It is the limited wait time that makes this problem solvable where the general halting problem is not.

Challenge Work

1. This is quite a complex problem. Students might be guided to an automata theory textbook for additional help. The general ideas follow:

a. Assume that one binary string is placed on the first track and the second binary string is placed on the second track, with their left ends aligned. Then the head must simply move from left to right across the two strings, comparing pairs of symbols. If at any point they do not match, then the machine should go into a HALT-NO MATCH state. If the head reaches the right end and hits matching blanks on the two tapes, then it should go into a HALT-MATCH state.

b. Assume that the tape initially contains the first binary string, a separating blank, and then the second binary string. The machine must move back and forth comparing corresponding elements in the two strings. Markers will need to be used to indicate positions that have already been compared. As before, if corresponding digits don't match then the machine should go into a HALT-NO_MATCH state. If, on the other hand, every digit matches and the strings are the same length, then the machine should go into a HALT_MATCH state.

c. This is a fairly easy proof. One can simulate a one-track machine trivially on a two-track machine by setting up the rules to carry out the original algorithm on the first track while ignoring whatever is on the second track.

d. This is much harder than part (c), but the idea is to put the contents of the first track, followed by a separating blank, followed by the contents of the second track all on a single tape. The single-track machine then uses several steps to carry out each step of the multitrack machine, namely reading a single symbol from the left half, laying down a marker, running to the right to read the corresponding symbol from the second half, and resetting the symbols in each half appropriately. As in part (b), there will be much sweeping back and forth between the locations on the single tape that represent corresponding positions of the two tracks. In addition, if the two-track machine ever writes symbols to the left of the starting location, then the one-track machine must copy the second half of the tape over farther to the right to create additional room after the separating blank.

2. Students might be interested to read or view "Breaking the Code," the play based on Alan Turing's life. A comparison of its perhaps fictionalized account with more biographical accounts would be interesting. Also the 2014 movie "The Imitation Game" depicts Alan Turing's work during World War II.

Chapter 13: Simulation and Modeling

1. A two-dimensional spreadsheet is a two-dimensional computational model of a system. By entering the appropriate data and formulas, a spreadsheet can be used to model the behavior of a set of entities such as a payroll or a population of animals . It is a form of mathematical model in which the data and the formulae are related to each other by their position in the grid. By changing values within the grid you can begin to answer the "what if" type questions described in this chapter.

2. CAD, computer aided design, is a combination of hardware and software that enables engineers and architects to design everything from furniture to airplanes. An engineer can view a design from any angle with the push of a button, and can zoom in or out for close-ups and long-distance views. In addition, the computer keeps track of design dependencies so that when the engineer changes one value, all other values that depend on it are automatically changed accordingly. CAD is related to simulation and modeling because it models real systems, especially in construction.
CAM, computer aided manufacturing, is a type of computer application that helps automate a factory. Examples of CAM systems include real-time control, robotics, and materials requirements. CAM relates to the ideas in this chapter because applications like real-time controllers keep track of a factory's behavior, which is like modeling its behavior.

3. Provide students with Web-based leads to the topics of SIMULA, GPSS, and Simscript.

4. Answers may vary. In addition to the air resistance and the fact that the Earth is not a perfect sphere, additional inaccuracies contained in this mathematical model include outside gravitational forces (for example, from other planets or the moon). These factors shouldn't be included in the falling body model because their effect is relatively so small that it can be neglected. On the other hand, wind velocity and wind direction, which exerts a horizontal force on the falling object, could be a very significant factor and should not be left out of the model.

5. Provide students with leads to sources about random number generators. Often random number generators use functions based on the computer clock's current time since midnight measured in milliseconds. The likelihood that the random number generator would be started at the exact (to the millisecond) time of day on two different days is near zero, so the numbers generated in each case would be different.

6. This may be a good approximation to the statistical distribution of service times, but it will not be completely accurate. Many factors determine the time of service, such as the time of day (breakfast orders can be filled faster than lunch orders?), the skill level of the workers, the number of customers being served (may have to wait for the next batch of french fries), etc. Because this distribution is an assumption in the model, the more inaccurate it is, the less reliance we should place on the conclusions of the model. Just as the garbage-in-garbage-out principle says, the model is only as good as the information one puts into it.

7. a. If instead of a single waiting line, we have N waiting lines, one for each of the N servers, then we would need to keep track of the number of customers in each waiting line.

Customer arrival algorithm: If all servers are busy, then put this customer at the end of the shortest waiting line and increase the length of that line by 1.

Customer departure algorithm: Take the next customer out of the corresponding server's line and decrease that line size by 1

Main simulation: Set all N waiting line sizes to zero on initialization instead of just the single line.

b. If the waiting line had a maximum length of MAX

```
Customer arrival algorithm:
        If waiting line size = MAX, then
                New customer departs
Else
        If everyone is busy…
```

Customer departure algorithm: unchanged
Main simulation: unchanged

c. If the priority system is used

Customer arrival algorithm: when the new customer first arrives, assign a priority number to the customer. If everyone is busy, begin at the end of the waiting line and go forward to the first person with an equal or higher priority than new customer. Insert new customer into line behind this person.

Customer departure algorithm: unchanged
Main simulation: unchanged

8. No, this is not a realistic assumption. As any restaurant owner knows, arrivals peak at certain times of the day (e.g., lunch, dinner, after theater) and slows dramatically at other times. To implement this behavior we could use the current value of the simulation clock. We might use one statistical distribution from time T = 0.0 minutes to 240.0 minutes (7AM to 11AM) which models a relatively low arrival rate. Then when T passes 240 we could switch to a completely different statistical distribution that models the higher arrival rate we might see during the lunch hour.

9. a. Yes, age could be a factor as people of different ages might order quite different amounts of food, thus affecting the service time of each customer

b. Possibly. Male and female customers might order different amounts of food, which would have an effect on service time. However, this would have to be examined closely to see if that assumption is valid and should be included.
c. No. Height would not have any effect on customer waiting or service time.

10. As in the case of the fast-food restaurant, one could make observations over a few days' time at the bus station. Or one could examine past records of bus ticket sales. In any case the desired outcome would be something of the form
23% of passengers travel to B
16% of passengers travel to C
39% of passengers travel to D
22% of passengers travel to E

11. Most likely it would be a continuous model. There are two reasons: First, the mathematical and physical laws governing how particles are created and destroyed during collisions as a function of their speed, energy, and atomic structure is reasonably well understood. Most likely it could be described as a set of formal mathematical equations.

Second, the number of distinct events that occur in a high-speed particle accelerator is so massive, it would be enormously time consuming, in a computational sense, to try and simulate each and every event in the same way we did with the McBurgers model.

12. a. One day of activity in the model would take three million seconds to simulate, since 10^{14} computations are needed for an hour of activity and, assuming a 24-hour day, 24×10^{14} instructions need to be computed. On a machine with a computation speed of 800 MIPS (which is $800\times10^{6)}$ instructions per second,

(24×10^{14} instructions) / (800×10^{6} instructions/second) = 3,000,000 seconds

which is nearly 35 days! It takes longer to run the simulation than the actual amount of time being simulated.

b. We would need a computer that can process 8 million MIPS. To find this, we use algebra. We know that 5 minutes is $5\times60 = 300$ seconds. The algebraic problem, then, is

(24×10^{14} instructions) / (x instructions/second) = 300 seconds

Solving for x, we get $x = 8\times10^{12}$ instructions / second = 8,000,000 MIPS, or about 8 teraflops.

13. Answers will vary. In addition to color and scale to enhance and highlight aspects of data sets being studied, one could use animation, pie charts, bar graphs, three-dimensionality, sorting, and slideshows to visually enhance the output of a model to help clarify its interpretation.

14. Provide leads to sources about the social sciences, humanities, anthropology, sociology, and political science. For example, Jay W. Forrester's work on modeling the dynamics of complex systems, including populations, economies, and ecological systems, may be a good place to start.

15. Answers will vary according to what field the student is in. Sources should be provided to simulation models being used in both science fields and humanities fields.

Challenge Work

The objects in this model include departing planes and arriving planes. The state of the system includes the number of planes waiting to take off and the number of planes waiting to land. The events are plane arrivals and plane departures, which are really requests for landing and requests for takeoff. Assume a known distribution for arrivals and departures, plus a fixed time for landing a plane and a fixed time for a plane to take off. The model must take into account the fact that only one plane may use the runway at a time, and should keep track of the time a plane waits for arrival or departure. Also note that arrivals and departures, unlike the McBurgers simulation, are unrelated events, that is, an arrival does not trigger a later departure, and planes can arrive while other planes are departing.

Chapter 14: Electronic Commerce and Databases

1. Answers will vary. Possible sites that can be used include Amazon.com, Ebay.com, and Dell.com.

2. Answers will vary. Poor retail Web sites have cluttered or "noisy" presentation, unclear or inconsistent navigability, lack of features, lack of flexibility, do not let the customer know where they are in the sales process or what is coming next, etc.

3. Answers will vary depending on sites the student has recently visited. To aid students, consider providing instructions on how to access the cookies folder/file.

4.

ID	LastName	FirstName	Birthdate	PayRate	HoursWorked
116	Kay	Janet	3/29/1980	$16.60	94
165	Honou	Morris	6/9/1997	$6.70	53

5.
```
SELECT FirstName, LastName, PayRate
FROM Employees
ORDER BY PayRate;
```

6.

ID	PlanType
149	B2
149	A1
149	C2

7.
```
SELECT FirstName, LastName, HoursWorked, PlanType
FROM Employees, InsurancePolicies
WHERE HoursWorked < 100
AND ID = EmployeeID;
(The result will be the "null set" - no employee fits this description)
```

8. Examples of InsurancePlans tuples:

A1	Basic medical/single	83.54
A4	Major medical/family	256.80
B2	Health management/single	68.90

9.
```
SELECT LastName, FirstName, InsurancePolicies.PlanType, MonthlyCost
FROM Employees, InsurancePlans, InsurancePolicies
WHERE LastName = "Kay" And FirstName = "John"
AND ID = EmployeeID
AND InsurancePolicies.PlanType = InsurancePlans.PlanType;
```

10. SELECT *
FROM Doctor;

SELECT *
FROM Patient;

SELECT *
FROM ClinicAppointment;

11. SELECT Doctor.FirstName, Doctor.LastName, Patient.FirstName, Patient.LastName, HomePhone, CellPhone, AppTime
FROM Doctor, Patient, ClinicAppointment
WHERE Doctor.DoctorID = ClinicAppointment.DoctorID
AND Patient.PatientID = ClinicAppointment.PatientID
AND ClinicAppointment.AppDate = #2/28/2018#;

Doctor.FirstName	Doctor.LastName	Patient.FirstName	Patient.LastName	HomePhone	CellPhone	AppTime
Anne	Davis	Gail	Perez	333-777-1212	333-410-7777	8:30:00 AM
Estelle	Villanueva	Gordon	Zhang	332-555-9999	332-217-4321	10:30:00 AM

12. SELECT FirstName, LastName, AppTime
FROM Doctor, ClinicAppointment
WHERE Doctor.DoctorID = ClinicAppointment.DoctorID
AND ClinicAppointment.AppDate=#2/27/2018#
AND Doctor.DoctorID ="DO1"
ORDER BY AppTime;

FirstName	LastName	AppTime
Vladimir	Yevgeny	11:30:00 AM
Vladimir	Yevgeny	1:00:00 PM

13. SELECT Patient.FirstName, Patient.LastName, AppTime, AppDate, Doctor.FirstName, Doctor.LastName
FROM Patient, Doctor, ClinicAppointment
WHERE Doctor.DoctorID = ClinicAppointment.DoctorID
AND Patient.PatientID=ClinicAppointment.PatientID
AND Patient.PatientID="PA3"
AND AppDate=#2/27/2018#;

Patients.FirstName	Patients.LastName	AppTime	AppDate	Doctors.FirstName	Doctors.LastName
DuWayne	Martin	1:00:00 PM	2/27/2018	Vladimir	Yevgeny

14. High blood pressure seems to be a small risk factor because there are only small differences in the percentages between low and high blood pressure individuals who otherwise classify the same on age and obesity. Age plays a slightly higher part because the percentage difference between the lows in either age classification is small and so is the percentage difference between the highs in either age classification. Obesity is the greatest risk factor, causing a split between lows and highs regardless of age or blood pressure.

15. a. The functions to use are: MIN, MEDIAN, AVERAGE, MAX
b. After the chart is created, right-click in the chart area and choose Select Data. Then click Switch Row/Column. Then Edit the Legend Series to change the names to Site 1, Site 2, etc. The result should now agree with Figure 10.9. c. Student should be able to just follow the instructions in the problem statement. Results should agree with the R results (with perhaps fewer decimal places) and, because the matrix is symmetric, only the lower triangular portion is shown.

Challenge Work

The appointment two days later would be March 1, 2018, because 2018 is not a leap year so there is no February 29. If you try to enter an appointment of the form

DO1 PA03 2/29/2018 9:00 AM Revisit chest pain

you will get an error message:

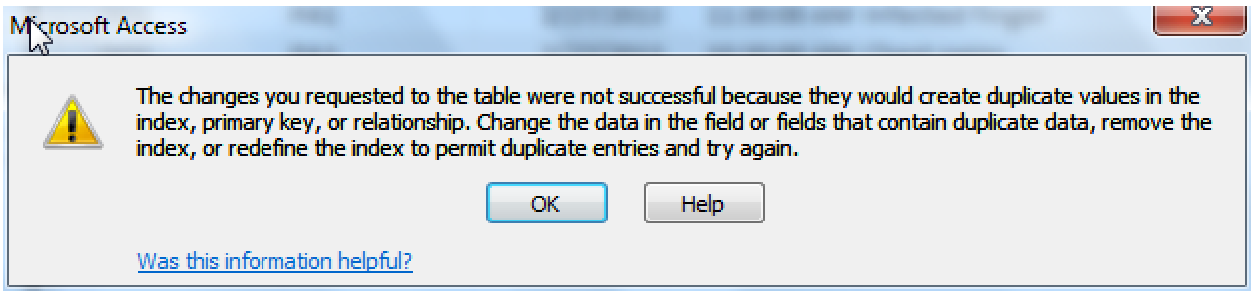
‘The value you entered does not match the Date/Time data type in this column’.

The system is enforcing the data integrity rule. This is the advantage of using the Date/Time data type rather than just Text. But this is not the major problem

If you try to insert

DO1 PA03 3/01/2018 9:00 AM Revisit chest pain

you get the message shown below. Why?

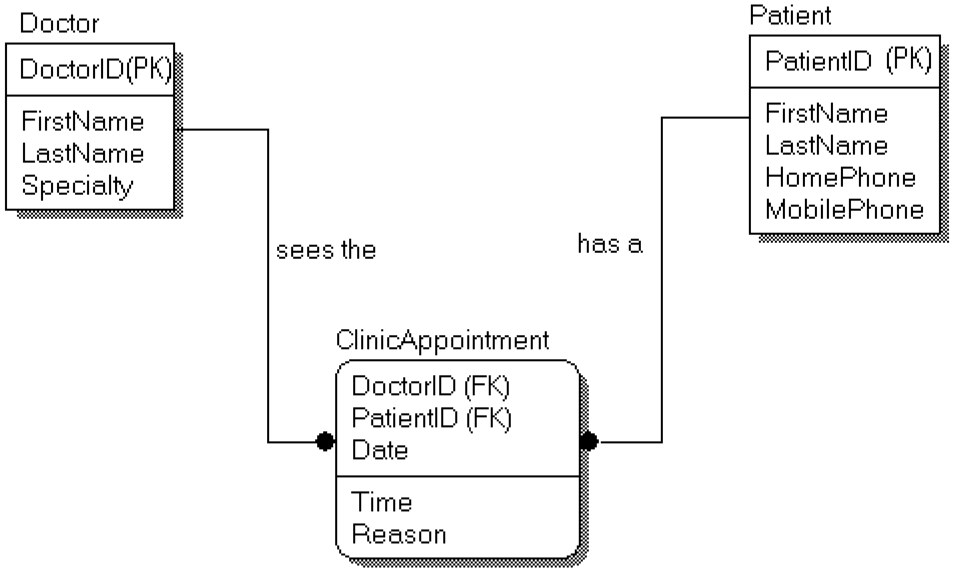


Microsoft Access

The changes you requested to the table were not successful because they would create duplicate values in the index, primary key, or relationship. Change the data in the field or fields that contain duplicate data, remove the index, or redefine the index to permit duplicate entries and try again.

With the current database design, the composite key DoctorID/PatientID is the primary key of the ClinicAppointment table, which means it should uniquely identify the tuple. Therefore there cannot be two different tuples with the same DoctorID/PatientID pair.

You need to modify the design of the database to look as follows.



Then you can add the new record. The SQL query is

```
SELECT *
FROM ClinicAppointment
ORDER BY AppDate, AppTime;
```

And the result is

ClinicAppointment

DoctorID	PatientID	AppDate	AppTime	Reason
DO1	PA1	2/27/2018	11:30:00 AM	Infected finger
DO1	PA3	2/27/2018	1:00:00 PM	Chest pains
DO4	PA2	2/28/2018	8:30:00 AM	Prenatal checkup
DO3	PA4	2/28/2018	10:30:00 AM	Poison ivy
DO1	PA3	3/1/2018	9:00:00 AM	Revisit chest pain

Chapter 15: Artificial Intelligence

1. a. For example: Evan asked Hank when the assignment is due because he had forgotten. Who had forgotten, Evan or Hank?

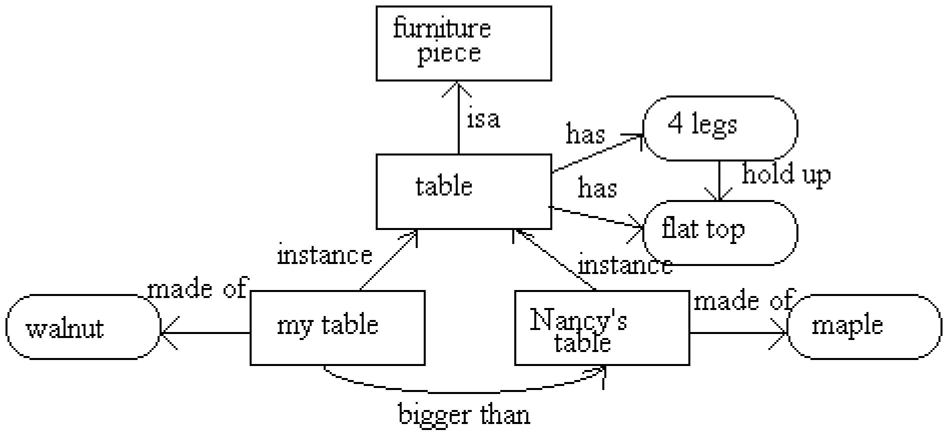
Answer: Evan

b. Evan asked Hank when the assignment is due but he had forgotten. Who had forgotten, Evan or Hank?

Answer: Hank

2. a. Jeremiah is a bullfrog.
b. All bullfrogs are green.

3. For example,

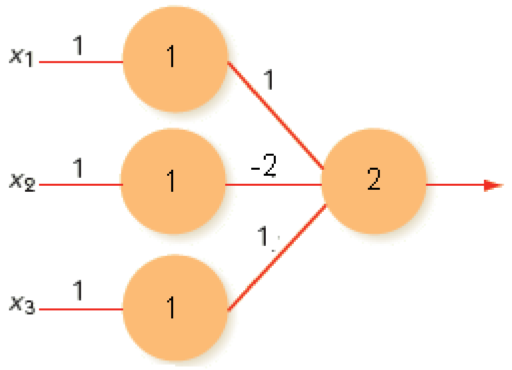


4. a. McIntosh is an apple.
All apples are red
All bananas are yellow.
Apples are fruit.
All bananas are fruit
All fruit is food
All meat is food
All food is eaten.

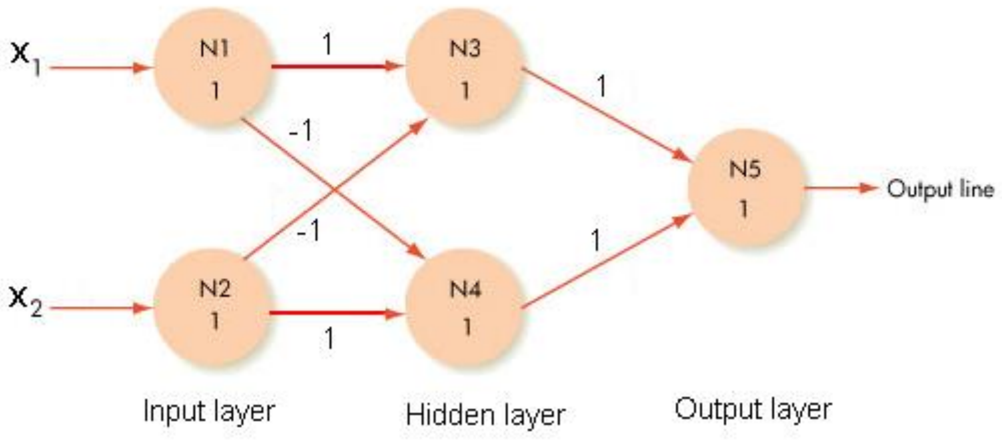
b. McIntosh is an apple, so it is red, it is a fruit and a food, and is eaten.
All apples are food and are eaten.
All bananas are food and are eaten.
All fruit is eaten.
All meat is eaten.

5. If $x_2 = 1$ and $x_4 = 1$, then N3 will fire, regardless of the values for x_1 and x_3.

6.



7. The following network produces results equal to the XOR truth table.



8. Provide leads to sources about mobile devices or tablet PC's, and pattern recognition software.

9. Students should be able to find material on the Web, but will have to distinguish between natural phenomena and artificial attempts to mimic natural phenomena.

10. For example,

How do you go about processing a loan?
What do you look for on the application?
What makes you suspicious that a loan applicant should not be granted the loan?
What do you look for to make you feel that the applicant should be granted the loan? Does one factor weigh more heavily than others in your decision?

11. The parse method described in Chapter 11 corresponds to backward chaining; it starts with the result of applying productions (similar to rules in knowledge-based systems) and tries to work back to the goal symbol (similar to assertions in knowledge-based systems).

12. Yes. Adopt the following notation:

A: hero is a spy
B: heroine is an interpreter
C: one scene should take place in Berlin
D: one scene should take place in Paris
E: heroine must speak English
F: heroine must speak Russian
G: there can be no car chase
H: there can be no crash scene
I: the hero is European
J: the hero must speak French

The assertions and rules can be symbolized as follows:

A
B
A $\rightarrow$ C and D
B $\rightarrow$ E
B $\rightarrow$ F
C $\rightarrow$ G
G $\rightarrow$ H
C $\rightarrow$ I
D $\rightarrow$ J

Since A is true and A implies C and D, both C and D are true. Since D is true and D implies J, then J is true. Since C is true and C implies G, then G is true. Since G is true and G implies H, then H is true. Therefore both J (Hero must speak French) and H (there can be no crash scene) are true.

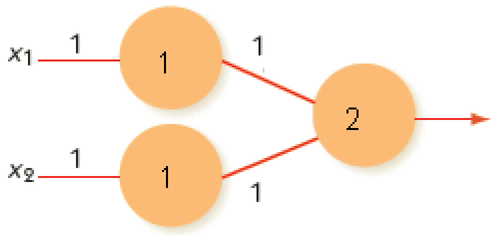
13. No; if the hero is American, then he is not European, which means that I is not true. But since C is true and C implies I, then I is true. I cannot be both true and false.

14. Freeware versions of Prolog are available, such as the Toy Prolog program at www.csse.monash.edu.au/~ lloyd/tildeLogic/Prolog.toy

Challenge Work

1. This is fairly heavy-duty philosophical stuff, so may be of interest only to your best students.

2. a.



b. The complete table is

w_1	w_2	θ	x_1	x_2	y	t	$\alpha(t - y)$	w_1'	w_2'	θ'
0.6	0.1	0.5	0	0	0	0	0	0.6	0.1	0.5
0.6	0.1	0.5	0	1	0	0	0	0.6	0.1	0.5
0.6	0.1	0.5	1	0	1	0	-0.2	0.4	0.1	0.7
0.4	0.1	0.7	1	1	0	1	0.2	0.6	0.3	0.5
0.6	0.3	0.5	0	0	0	0	0	0.6	0.3	0.5
0.6	0.3	0.5	0	1	0	0	0	0.6	0.3	0.5
0.6	0.3	0.5	1	0	1	0	-0.2	0.4	0.3	0.7
0.4	-.3	0.7	1	1	0	1	0.2	0.6	0.5	0.5
0.6	0.5	0.5	0	0	0	0	0	0.6	0.5	0.5
0.6	0.5	0.5	0	1	0	0	0	0.6	0.5	0.5
0.6	0.5	0.5	1	0	1	0	-0.2	0.4	0.5	0.7
0.4	0.5	0.7	1	1	1	1	0	0.4	0.5	0.7
0.4	0.5	0.7	0	0	0	0	0	0.4	0.5	0.7
0.4	0.5	0.7	0	1	0	0	0	0.4	0.5	0.7
0.4	0.5	0.7	1	0	0	0	0	0.4	0.5	0.7

After the heavy line, the network has learned a solution. The bottom four rows of the table indicate that the network is completely trained.

3. Students should be able to find lots of information on these topics.

Chapter 16: Computer Graphics and Entertainment: Movies, Games, and Virtual Communities

1. The text suggests Wikipedia as a starting point, but you should expect further research on the students' part. A good article to look at would be http://www.pcmag.com/encyclopedia_term/0,2542,t=graphics+pipeline&i=43933,00.asp

2.

	vertex	*x*	*y*	*connected to*			
(origin)	v_1	0	0	v_2	v_4	v_5	
	v_2	0.5	1.8	v_1	v_3	v_5	
	v_3	2.2	2.0	v_2	v_4	v_5	
	v_4	2.6	0.5	v_1	v_3	v_5	
	v_5	1	1	v_1	v_2	v_3	v_4

3. a. Since the motion takes place over 10 seconds, and the standard frame rate for video is 30 frames per second, we must produce a total of 300 frames. To create each new frame we must move 100,000 vertices, and each vertex requires a matrix multiplication with 28 arithmetic operations per vertex. Thus, the total number of operations required to carry out this operation is:

10 seconds $\times$ 30 frames/sec $\times$ 100,000 vertices/frame $\times$ 28 operations/vertex
= 840,000,000 arithmetic operations = 8.4×10^8 arithmetic operations

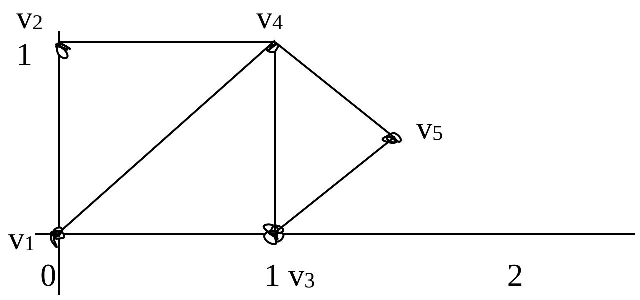
b. At 50 million operations/sec, 5×10^7, the time required to carry out this translation operation will be:
= $(8.4 \times 10^8 / 5 \times 10^7)$ = 16.8 seconds

4. In each second of real time we need to generate 30 frames, with each frame containing 250,000 vertices. This will require the computation of

30 frames $\times$ 250,000 vertices/frame $\times$ 28 operations/vertex
= 2.1×10^8 operations in each second of real time

Thus we need a GPU that can execute at a rate of at least 210 Mflops

5.



6. We need to move the object three units in the x-direction, five units in the y-direction, and zero units in the z-direction. Since the motion lasts for two seconds, we need to have a total of 60 frames, which will require 59 separate movements starting from the initial position in the first frame. Using the translation matrix model shown in Figure 16.6, here is a matrix to accomplish this motion:

$$\begin{vmatrix} 1 & 0 & 0 & 3/59 \\ 0 & 1 & 0 & 5/59 \\ 0 & 0 & 1 & 0/59 \\ 0 & 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0.05084 \\ 0 & 1 & 0 & 0.08475 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

7. The correct rotation matrix is:

$$\begin{vmatrix} \cos Ø & -\sin Ø \\ \sin Ø & \cos Ø \end{vmatrix}$$

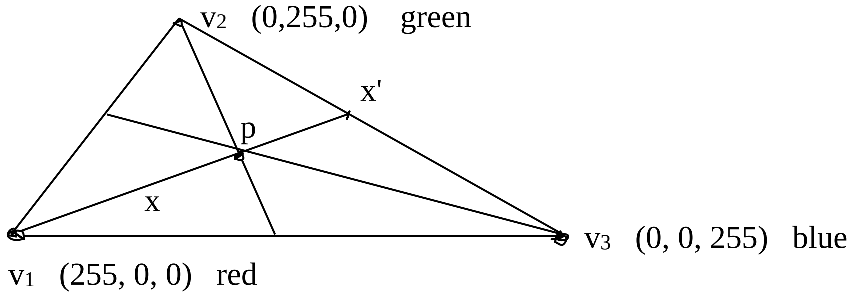
8. The correct reflection matrix to reflect an object around the y-axis is:

$$\begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$$

9. Point A would be used to raise and/or lower the entire arm as a single unit. much like raising a stick. This control point could be used to raise the arm high enough so that the glass can reach the mouth. Point B would be used to bend the elbow, while the rest of the arm, from elbow to shoulder remains motionless. So, once the arm is in the correct position, this control point could be used to raise the glass to the level of the mouth. Finally, Point C is used to bend the wrist and align the angle of the hand. This control point could be used to align the glass to properly meet the mouth.

10. A flight simulation package must absolutely be a real-time graphics package if it is intended to train pilots. The response rate of the simulator display must be identical to the real-life response rate of the actual airplane controls, otherwise it would not be an effective learning and teaching tool. So, in many ways, the graphical software routines used in an airline simulator would be similar in structure to a video game software package, and a flight simulator would also potentially need to sacrifice realism to obtain the necessary display speed.

11. There are many different coloring algorithms. One popular approach to shading in a triangular face is to color each pixel inside the face based on its normalized distance from each of the three vertices of the triangle. For example, given the triangle shown in the question:



Point p is x units from vertex v_1, while the total length of the line from vertex v_1 to the line connecting vertices v_2 and v_3 is x'. Since the red contribution of vertex v_1 is 255 we could simply multiply 255 times the normalized distance of p from v_1, namely [1 - (x / x')]. When p is located on vertex v_1 this term become 1 - 0 = 1, and the red contribution is 1 * 255 = 255, the full amount. When p moves to the end of that line, point x', this term becomes 1 - 1 = 0, and the red contribution becomes 0 * 255 = 0, or nil. Thus the red component of the color of point p diminishes linearly with the distance of point p from vertex v_1. We now do the same thing with point p and its distance from first v_2 and then v_3. The end result is a three-tuple (i, j, k), which gives the red, green, and blue components, resulting in a single color, and that is the color we assign to pixel p.

12. Formal mathematical explanation is not called for, but students should be able to give an informal rule of thumb.

13. a. We would have a circle of radius 1 whose center is at (-6, 0)

b. We would have a circle of radius 1 whose center is at (-2, 0)

c. No change. Two reflections about the same mirror line will produce the original image.

Challenge Work

While there is plenty of information available on these topics, the trick is to find explanations at a level suitable for students to understand.