**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 *ci* is not equal to 0 or *i* < 0

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

 Step 11: If *i* > 0 then print *c*i*c*i-1 . . . *c*0

**7**. Assume that *a* has *n* digits an-1, … , a0, and *b* has *m* digits, bm-1, … , b0, 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 an+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 bm+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 – 4*ac* < 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 x 1025 different paths. The computer can analyze 10,000,000, or 107, paths per second. The number of seconds required to check all possible paths is about 1.5 x 1025/107, or about 1.5 x 1018 seconds. That’s roughly 1012 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 ≥ 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 (*ai*) is smaller than the lower, we automatically borrow from the next column. We can do this by subtracting one from the *ai+1* value of the column to the left. If the *ai+*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 *bi* < *ai* then

 Step 4: Set *ci* equal to *ai - bi*

 Otherwise (*bi > ai*)

 Step 5: Set *ci* equal to (*ai +* 10) – *bi* and replace *ai+*1 with *ai+*1 – 1

 (This amounts to a borrow of 1 from *ai*+1 which adds 10 to *ai*)

 Step 6: Add 1 to *i* (moving us one column to the left)

 Step 7: Print out the final answer *cm-*1*cm-*2 . . . *c0*

Step 8: Stop.

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