CMPS 10 Winter 2011 - Homework Assignment 1

Problems:
Chapter 1 (p. 34): 3, 4, 5, 7ab, 8, 9

Chapter 1

3.) Trace through the decimal addition algorithm of Figure 1.2 using the following input values:
   \( m = 3 \)
   \( a_2 = 1, a_1 = 4, a_0 = 9 \)
   \( b_2 = 0, b_1 = 2, b_0 = 9 \)

At each step, show the values for \( c_3, c_2, c_1, c_0, \) and \( \text{carry} \).

Solution:
Note: “undef” stands for “undefined” and indicates an unset value.

* Start at step 1. \( c_3 = \text{undef} \) \( c_2 = \text{undef} \) \( c_1 = \text{undef} \) \( c_0 = \text{undef} \) \( \text{carry} = 0 \)
* Do steps 2 and 3: \( c_3 = \text{undef} \) \( c_2 = \text{undef} \) \( c_1 = \text{undef} \) \( c_0 = \text{undef} \) \( \text{carry} = 0 \)
* Step 4 \((i = 0)\): \( c_3 = \text{undef} \) \( c_2 = \text{undef} \) \( c_1 = \text{undef} \) \( c_0 = 18 \) \( \text{carry} = 0 \)
* Step 5 \((i = 0)\): \( c_3 = \text{undef} \) \( c_2 = \text{undef} \) \( c_1 = \text{undef} \) \( c_0 = 8 \) \( \text{carry} = 1 \)
* Step 6 \((i = 1)\): \( c_3 = \text{undef} \) \( c_2 = \text{undef} \) \( c_1 = \text{undef} \) \( c_0 = 8 \) \( \text{carry} = 1 \)
* Step 4 \((i = 1)\): \( c_3 = \text{undef} \) \( c_2 = \text{undef} \) \( c_1 = 7 \) \( c_0 = 8 \) \( \text{carry} = 1 \)
* Step 5 \((i = 1)\): \( c_3 = \text{undef} \) \( c_2 = \text{undef} \) \( c_1 = 7 \) \( c_0 = 8 \) \( \text{carry} = 0 \)
* Step 6 \((i = 2)\): \( c_3 = \text{undef} \) \( c_2 = \text{undef} \) \( c_1 = 7 \) \( c_0 = 8 \) \( \text{carry} = 0 \)
* Step 4 \((i = 2)\): \( c_3 = \text{undef} \) \( c_2 = 1 \) \( c_1 = 7 \) \( c_0 = 8 \) \( \text{carry} = 0 \)
* Step 5 \((i = 2)\): \( c_3 = \text{undef} \) \( c_2 = 1 \) \( c_1 = 7 \) \( c_0 = 8 \) \( \text{carry} = 0 \)
* Step 6 \((i = 3)\): \( c_3 = \text{undef} \) \( c_2 = 1 \) \( c_1 = 7 \) \( c_0 = 8 \) \( \text{carry} = 0 \)
* Step 7: \( c_3 = 0 \) \( c_2 = 1 \) \( c_1 = 7 \) \( c_0 = 8 \) \( \text{carry} = 0 \)
* Step 8: Print out 0178.

4.) Modify the decimal addition algorithm of Figure 1.2 so that it does not print out nonsignificant leading zeroes; that is, the answer to question 3 would appear as 178 rather than 0178.

Solution:
There are several possible solutions to this problem. Here is one possible version:

Remove the current step 8, and replace it with the following steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Set the value of ( i ) to ( m ).</td>
</tr>
<tr>
<td>9</td>
<td>While ( c_i = 0 ) and ( i \geq 0 ) do step 10.</td>
</tr>
<tr>
<td>10</td>
<td>Subtract 1 from ( i ), moving one digit to the right.</td>
</tr>
<tr>
<td>11</td>
<td>If ( i \geq 0 ) then print ( c_0 c_{i-1} \ldots c_0 ).</td>
</tr>
</tbody>
</table>
5.) Under which conditions would the well-known quadratic formula

\[ \text{Roots} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

not be effectively computable?  (Assume you are working with real numbers).

Solution:

There are two circumstances in which the quadratic formula is not effectively computable.
The first is if \( a = 0 \), because we cannot divide by zero.
The second is if \( b^2 - 4ac < 0 \), since we cannot take the square root of a negative number and obtain an answer in the real numbers.

7.) The following is Euclid’s 2,300-year-old algorithm for finding the greatest common divisor of two positive integers \( I \) and \( J \):

<table>
<thead>
<tr>
<th>Step</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get two positive integers as input. Call the larger value ( I ) and the smaller value ( J ).</td>
</tr>
<tr>
<td>2</td>
<td>Divide ( I ) by ( J ) and call the remainder ( R ).</td>
</tr>
<tr>
<td>3</td>
<td>If ( R ) is not 0, then reset ( I ) to the value of ( J ), reset ( J ) to the value of ( R ), and go back to step 2.</td>
</tr>
<tr>
<td>4</td>
<td>Print out the answer, which is the value of ( J ).</td>
</tr>
<tr>
<td>5</td>
<td>Stop.</td>
</tr>
</tbody>
</table>

a.) Go through this algorithm using the input values 20 and 32. After each step of the algorithm is completed, give the values of \( I \), \( J \), and \( R \). Determine the final output of the algorithm.

Solution:

* Step 1: \( I = 32 \), \( J = 20 \), \( R = \text{undef} \)
* Step 2: \( I = 32 \), \( J = 20 \), \( R = 12 \)
* Step 3 (Set \( I \) to \( J \), and \( J \) to \( R \)): \( I = 20 \), \( J = 12 \), \( R = 12 \)
* Step 2: \( I = 20 \), \( J = 12 \), \( R = 8 \)
* Step 3 (Set \( I \) to \( J \), and \( J \) to \( R \)): \( I = 12 \), \( J = 8 \), \( R = 8 \)
* Step 2: \( I = 12 \), \( J = 8 \), \( R = 4 \)
* Step 3 (Set \( I \) to \( J \), and \( J \) to \( R \)): \( I = 8 \), \( J = 4 \), \( R = 4 \)
* Step 2: \( I = 8 \), \( J = 4 \), \( R = 0 \)
* Step 4: Print \( J = 4 \).

b.) Does the algorithm work correctly when the two inputs are 0 and 32? Describe exactly what happens, and modify the algorithm so that it gives an appropriate error message.

Solution:

In step 1, \( I = 32 \) and \( J = 0 \). In step 2, the algorithm attempts to perform the calculation \( 32/0 \), which produces a divide-by-zero error. The simplest fix is to add a step between 1 and 2 (and renumber the subsequent steps). This new step 2 would be:
8.) A salesperson wants to visit 25 cities while minimizing the total number of miles she has to drive. Because she has studied computer science, she decides to design an algorithm to determine the optimal order in which to visit the cities to (1) keep her driving distance to a minimum, and (2) visit each city exactly once. The algorithm that she has devised is the following:

The computer first lists all the possible ways to visit the 25 cities and then, for each one, determines the total mileage associated with that particular ordering. (Assume that the computer has access to a road map that provides the distances between all cities). After determining the total mileage for each possible trip, the computer searches for the ordering with the minimum mileage and prints out the list of cities on that optimal route, that is, the order in which the salesperson should visit her destinations.

If a computer could analyze 10,000,000 separate paths per second, how long would it take to determine the optimal route for visiting these 25 cities? On the basis of your answer, do you think that this is a feasible algorithm? If it is not, can you think of a way to obtain a reasonable solution to this problem?

Solution:
Since there are 25 cities that need to be visited, there are 25! possible paths that need to be considered. A quick calculation tells us that 25! = 1.551121 x 10^{25} paths. The computer can analyze 10,000,000 (10^7) paths per second, it will take 1.551121 x 10^{25}/10^7 = 1.551121 x 10^{18} seconds. This is nearly 50 billion years, which is not a feasible algorithm.

9.) One way to do multiplication is by repeated addition. For example, 47 x 25 can be evaluated as 47 + 47 + 47 + … + 47 (25 times). Sketch out an algorithm for multiplying two positive numbers a and b using this technique.

Solution:
There are several possible algorithms that will work. Here is one version which assumes we know how to add two multiple digit numbers together. This is a safe assumption, since we already have an algorithm from the book to do that.

<table>
<thead>
<tr>
<th>Step</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Take a and b as input.</td>
</tr>
<tr>
<td>2</td>
<td>Set c equal to 0, and set i equal to b.</td>
</tr>
<tr>
<td>3</td>
<td>While i is greater than 0, repeat steps 4 and 5.</td>
</tr>
<tr>
<td>4</td>
<td>Set the value of c to be c + a.</td>
</tr>
<tr>
<td>5</td>
<td>Subtract 1 from i.</td>
</tr>
<tr>
<td>6</td>
<td>Print out the final answer to c and STOP.</td>
</tr>
</tbody>
</table>