Assignment #1: Number Representation on a Computer, Loss of Precision, Systems of Linear Equations, Echelon Form

Due date: Wednesday, January 30, 2019 (9:15am)

Name: ____________________________________________

Section Number
Assignment #1: Number Representation on a Computer, Loss of Precision, Systems of Linear Equations, Echelon Form

Due date: Wednesday, January 30, 2019 (9:15am)
For full credit you must show all of your work.

1. (4 points) Suppose you have 8 binary bits (one byte) available to encode a number on a computer. How many different numbers could you represent, at most, using those 8 bits? How many different numbers could you represent, at most, using 16 bits (two bytes)?

2. For each of the three different encoding schemes listed below, please (a) specify each of the different numbers that could be exactly represented, and (b) characterize the numbers that cannot be exactly represented. The example below shows the type of information I am looking for in your answer to this question.

Example: 3-bit unsigned integers:
(a) Each 0 or 1 represents a binary digit. Three binary digits abc (where each of a, b, c is either 0 or 1) encode the number: \( a \times 2^2 + b \times 2^1 + c \times 2^0 \). A total of 8 different numbers can be represented using this encoding. They are: 000\(_2\) = 0; 001\(_2\) = 1; 010\(_2\) = 2; 011\(_2\) = 3; 100\(_2\) = 4; 101\(_2\) = 5; 110\(_2\) = 6; 111\(_2\) = 7
(b) Numbers less than zero would be out-of-range due to underflow; numbers greater than 7 would be out-of-range due to overflow; real numbers in the range (0-1), (1-2), etc., would be within range, but not exactly representable using this encoding. For example, the real number 3.12 would need to be truncated and stored as 011\(_2\) = 3 using this encoding.

a) (6 points) 4-bit signed integers
b) (8 points) 4-bit unsigned fractional numbers \( x \), where \( 0 \leq x < 1 \)
c) (8 points) 4-bit fixed-point fractional numbers, in which 2 of the bits are used to represent the integer part and 2 of the bits are used to represent the fractional part

3. (8 points) Show how you could encode 16 different unsigned numbers in a simplified floating point format using 4 bits. Use 2 bits to encode four different values of a biased exponent (2\(\text{e} \)) where e is encoded using 2 binary bits. (In this small example we won’t be reserving the largest and smallest exponents as flags for special handling.) Use 2 bits to encode a normalized mantissa of the form \( 1.f \), where the f is the part that is encoded by your two bits. Enumerate the 16 different binary encodings and their decimal equivalent. How does the set of numbers you obtain using this encoding differ from the set of numbers you obtained in your answer to question 2c?

4. (4 points) Suppose you are writing a program to manage banking data, and that you are restricted to working with a 4-byte data type (32 bits). Explain why it would be preferable to use a 32-bit integer to represent money in units of pennies rather than to use a floating point variable. Assume that the only transactions you need to record are deposits and withdrawals. Hint: consider issues related to numerical error and precision in number representation.

5. (6 points) Which of these numbers can be represented exactly on a computer (using the IEEE standard binary encoding for real numbers)? (please circle all that apply)
(a) 1 (b) 0.1 (c) 0.5
6. (5 points)
(a) Single precision floating point numbers are represented using 32 bits, 23 of which encode the normalized mantissa. Use the Matlab command `realmax('single')` to find the largest positive real number that can be represented using 32 bits.

(b) Double precision numbers are represented using 64 bits. Use the Matlab command `realmax('double')` to find the largest positive real number that can be represented using 64 bits, 52 of which encode the normalized mantissa.

(c) Why do you think many programs for things like image processing, self-driving cars, etc., are written to work with double precision numbers? Is it because the additional bits are needed in order to be able to work with larger numbers? Or because the additional bits allow numbers to be stored on the computer with greater precision?

7. Use Matlab to run the code shown below, which attempts to numerically compute 
\[ y = \frac{d}{dx} \sin(x) \] using the classical definition 
\[ f'(x) = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \] for increasingly smaller values of \( h \).

```matlab
h = 1;
x = 0.5;
for i = 1:30
    h = h/4;
    y(i) = (sin(x + h) - sin(x))/h;
    error(i) = abs(cos(x) - y(i));
end
```

a) (2 points) As \( h \) gets smaller, what trend do you observe in the value of `error`?

b) (3 points) As \( h \) gets smaller, what trend do you observe in the value of \( y \) (the approximated answer)?

c) (5 points) Edit the provided code to compute the value of the numerator: \( (\sin(x + h) - \sin(x)) \) independently, so that you can interrogate it explicitly. What do you notice about the computed value of the numerator as the value of \( h \) gets small? (Hint: pay attention to the number of zeros you see. Where do the zeros come from?) How do you reconcile the errors you encounter in computing \( (\sin(x + h) - \sin(x)) \) with the fact that it is possible for the computer to exactly represent values of \( h \) that are much smaller than \( 10^{-16} \)?

d) (4 points) What is the essential reason for the failure of this algorithm? Please be explicit; general statements such as “There is numerical error” will not receive credit. Obviously there is numerical error. Where is it coming from? Why does it occur?

Please note: you do not need to turn in a printout from running the Matlab code.

8. Consider the following system of linear equations:
\[
\begin{align*}
2x_1 + 3x_2 + x_3 &= 11 \\
4x_1 - 2x_2 + 3x_3 &= 9 \\
x_1 + x_2 - x_3 &= 0
\end{align*}
\]

a) (3 points) Express this system as an augmented matrix

b) (9 points) Show, step-by-step, how to reduce this augmented matrix and solve the system.
9. (9 points) Give an example of a system of linear equations in three variables that has:
   a) one unique solution
   b) no solution
   c) an infinite number of solutions
   In each case, express your answer in the form: \( a_1x_1 + a_2x_2 + a_3x_3 = b_1 \), etc., where each of the \( a_i \) and \( b_\) are non-zero.

10. (8 points) Suppose \( a, b, c, \) and \( d \) are non-zero constants. What relationship must hold among the numbers \( a, b, c, \) and \( d \) in order to guarantee that the system below is consistent for all possible values of \( f \) and \( g \)?
    \[
    ax_1 + bx_2 = f \\
    cx_1 + dx_2 = g
    \]

11. (4 points) A system of linear equations that has fewer equations than unknowns is sometimes called an underdetermined system. Can such a system ever have a unique solution? Explain.

12. (4 points) A system of linear equations that has more equations than unknowns is sometimes called an overdetermined system. Can such a system ever be consistent? Explain.