CMPS 251-Numerical Computing
Assignment 2
Due Thursday, October 1, 2015

Reading Material:
- Cheney & Kincaid: Sections 1.3 and 1.4

Notes: You are encouraged in work individually on the assignment. Piazza can be used to ask questions (without requesting a solution !!).

Problem 1 *Decimal to Binary*

Write a user-defined MATLAB function that converts real numbers in decimal form to binary form. Your code should take a decimal number as an input and generate the binary representation in a vector of 30 elements, the first 15 elements are reserved for the integer part and the last 15 for the decimal part. Your code should also check that the integer part of the input number is smaller than $((111111111111111)_2 = (32767)_{10})$ and output an error otherwise. The code should be structured as follows

```
function y = dec2binfull (x)
%x:  input decimal value
%y:  converted binary value
% Your code for checking that the input is less than the maximum value goes here
...
% Your code for converting the integer part goes here
...
% Your code for converting the decimal part goes here
...
```

The following inbuilt functions would be useful in this problem: `floor()`, `error()`, `zeros()`, `break`.

Example:
`y=dec2binfull(52.234375)` yields `y=000000000110100001111000000000)`.

Problem 2 *Function Reformulation*

Part 1-Loss of Significance
Consider the function $f(x) = \frac{1-\cos(x)}{\sin(x)}$

a) Use the decimal format with six significant digits (apply rounding at each step) to calculate $f(x)$ for $x = 0.007$ using a calculator.

b) Use MATLAB (use `format long`) to calculate the value of $f(x)$. Consider this to be the true value, and calculate the true relative error, due to rounding, in the value of $f(x)$ that was obtained in part (a).

c) Discuss why computing the function f near 0 would cause a problem. Arrange the formula to avoid loss of significance, if any?

Part 2-Overflow

For very large values of x and y, the following function would cause an overflow

$$f(x) = \sqrt{x^2 + y^2}$$

Reformulate this function to avoid any overflow.

Problem 3 *New Rounding Scheme!!!!*

A CMPS 251 student enjoyed the derivation of the rounding and chopping errors in a general floating point representation. He decided to come up with a rounding scheme and calculate the resulting rounding error. He defined the following rounding scheme.

Consider a real number x represented in base β system as follows

$$x = (.d_1 d_2 d_3 \ldots d_t d_{t+1} \ldots) \cdot \beta^e = \left(\sum_{i=1}^{\infty} d_i \beta^{-i} \right) \cdot \beta^e, \quad d_1 \neq 0$$

The "FunkyRound" scheme is given as

$$\text{fl}(x) = \begin{cases} (.d_1 d_2 \ldots [d_t + 1]) \cdot \beta^e = \left(\sum_{i=1}^{t} d_i \beta^{-i} + \beta^{-t} \right) \cdot \beta^e & \text{if } d_{t+1} < \beta/2 \\ (.d_1 d_2 \ldots d_t) \cdot \beta^e = \left(\sum_{i=1}^{t} d_i \beta^{-i} \right) \cdot \beta^e & \text{if } d_{t+1} > \beta/2 \\ (.d_1 d_2 \ldots d_{t-1} 0) \cdot \beta^e = \left(\sum_{i=1}^{t-1} d_i \beta^{-i} \right) \cdot \beta^e & \text{if } d_{t+1} = \beta/2 \end{cases}$$

Derive the "FunkyRound" maximum rounding error, Comment?

Problem 4 *Floating-Point Numbers*

Part 1
Identify the floating-point numbers that correspond to the following bit strings. Justify your answers

```
0 00000000 00000000000000000000000

1 00000000 00000000000000000000000

0 11111111 00000000000000000000000

1 11111111 00000000000000000000000

0 00000001 00000000000000000000000

0 10000001 01100000000000000000000

0 01111111 00000000000000000000000

0 01111011 10011001100110011001100
```

Part 2

Determine the numbers that have the following machine representations. Show all the steps

1)$[C705A700]_{16}$ 2)$[45607000]_{16}$ 3)$[494F96A0]_{16}$

Part 3

Determine the single precision machine representation of the following numbers. Show all the steps. Calculate the relative error between the given number and its machine representation if it cannot be represented *exactly* in single precision format.
1)64.37109375 2)0.234375 3)-9876.54321

Problem 5 *Floating-Point Operations*

Let $a = 0.23371258 \times 10^{-4}$, $b = 0.33678429 \times 10^2$, $c = -0.33677811 \times 10^2$. Assuming an 8-decimal-digit computer, determine the sum $s = a + b + c$ either as (1) $\mathrm{fl}(s) = \mathrm{fl}(\mathrm{fl}(a + b) + c)$ (2) $\mathrm{fl}(s) = \mathrm{fl}(a + \mathrm{fl}(b + c))$ Explain the discrepancy between the two answers.

For arbitrary machine numbers a, b, c on a computer with machine precision $\epsilon = 2^{-24}$, find a criterion on a, b, c for the result of (2) to be more accurate than the result of (1). Hint: compare bounds on the relative errors, neglecting higher-order terms in δ and assuming $a + b + c \neq 0$