Assignment 2

Objectives:

In this assignment, you will gain familiarity with:

- IEEE floating point representation

Submission:

- When creating your assignment document (i.e., your answers to this assignment), please,
    - Include the number of the question you are answering (e.g., Question 1.a. I) followed by your answer, keeping the questions in their original numerical order. Formatting your assignment document this way makes it a lot easier to mark. ☺
    - Add your full name and student number at the top of the first page of your document.
- Submit your document called **Assignment_2.pdf**, which must include your answers to all of the questions in Assignment 2.
- **If you write your answers by hand (as opposed to using a computer application to write them),** when putting your assignment document together, do not take photos (no .jpg) of your assignment sheets! Scan them instead!
  Better quality -> easier to read -> easier to mark! ☺

Due:

- Friday, Oct. 1 at 4pm on CourSys.
- Late assignments will receive a grade of 0, but they will be marked (if they are submitted before the solutions are posted on Monday) in order to provide feedback to the student.

Requirements:

- **Show your work** (as illustrated in lectures).

Marking scheme:

- This assignment will be marked as follows:
  - Questions 1 and 2 will be marked for correctness.
- The amount of marks for each question is indicated as part of the question.
- A solution will be posted on Monday after the due date.

---

1. [8 marks] Floating point conversion and Rounding.
   a. Represent the following numbers in IEEE floating point representation (single precision), clearly showing the effect of rounding on the `frac` (mantissa) if rounding occurs. Then express your final answer in binary and in hexadecimal form.

      I. 0.001111111_2

      II. 3.1416015625_{10}

      III. -0.9_{10}

      IV. $1/3_{10}$ (a third)

   b. Convert 0x4AEA4C1A from IEEE floating point representation (single precision) to a fractional decimal number (i.e., a real number).

c. Round the following binary numbers (rounding position is bolded – it is the bit at the 2^{-4} position -) following the rounding rules of the IEEE floating point representation.

   I. $1.001\mathbf{1}111_2$

   II. $1.100\mathbf{1}001_2$

   III. $1.011\mathbf{1}100_2$

   IV. $1.011\mathbf{0}100_2$

   For each of the above rounded binary numbers, indicate what type of rounding you performed and compute the value that is either added to or subtracted from the original number (listed above) as a result of the rounding process. In other words, compute the error introduced by the rounding process.

2. [12 marks] Creating hypothetical smaller floating-point representations based on the IEEE floating point format allows us to investigate this encoding scheme more easily, since the numbers are easier to compute and manipulate.

   Below is a table listing several fractional decimal numbers represented as 6-bit IEEE-like floating-point numbers (w = 6). The format of these 6-bit floating-point numbers is as follows: 1 bit is used to express for the sign (**s**), 3 bits are used to express **exp** (k = 3) and 2 bits are used to represent **frac** (n = 2), in the following order: **sign exp frac**.

   Complete the table (the same way as in Figure 2.35 in our textbook) then answer the questions below the table.

   Tip: Have a look at Figure 2.35 in our textbook, which illustrates a similar table for a hypothetical 8-bit IEEE-like floating-point format. This will give you an idea of how to complete the table. Also, Figure 2.34 displays the complete range of these 6-bit IEEE-like floating point numbers as well as their values between -1.0 and 1.0. This diagram may be helpful when you are checking your work.

		Exponent			Fraction		Value		
Description	**Bit representation**	**exp**	**E**	**2^E**	**frac**	**M**	**M 2^E**	**V**	**Decimal**
zero	0 000 00							0	0.0
Smallest positive denormalized	0 000 01								
	0 000 10								
Largest positive denormalized	0 000 11								
Smallest positive normalized	0 001 00								
	0 001 01								
	0 001 10								
	0 001 11								
	0 010 00								
	0 010 01								
	0 010 10								
	0 010 11								
One	0 011 00								1.0

	0 011 01								
	0 011 10								
	0 011 11								
	0 100 00								
	0 100 01								
	0 100 10								
	0 100 11								
	0 101 00								
	0 101 01								
	0 101 10								
	0 101 11								
	0 110 00								
	0 110 01								
	0 110 10								
Largest positive normalized	0 110 11								

+ Infinity		_	_	_	_	_	_		_
NaN		_	_	_	_	_	_	NaN	_

a. What is the value of the bias?

b. Consider two adjacent denormalized numbers. How far apart are they? Expressed this difference as a fractional decimal number (i.e., a real number).

c. Consider two adjacent normalized numbers with the **exp** field set to 001. How far apart are they? Expressed this difference as a decimal number.

d. Consider two adjacent normalized numbers with the **exp** field set to 010. How far apart are they? Expressed this difference as a decimal number.

e. Consider two adjacent normalized numbers with the **exp** field set to 011. How far apart are they? Expressed this difference as a decimal number.

f. Without doing any calculations, can you guess how far apart are two adjacent normalized numbers …

    a. with the **exp** field set to 100?

    b. with the **exp** field set to 101?

    c. with the **exp** field set to 110?

g. What is the "range" (not contiguous) of fractional decimal numbers that can be represented using this 6-bit IEEE-like floating-point representation?

h. What is the range of the normalized exponent **E** (**E** found in the equation $v = (-1)^s M 2^E$) which can be represented by this 6-bit IEEE-like floating-point representation?

i. Give an example of a fractional decimal numbers that cannot be represented using this 6-bit IEEE-like floating-point representation, but is within the "range" of representable values, which you expressed as your answer to Question g. above.

j. Give an example of a real number that would overflow if we were trying to represent it using this 6-bit IEEE-like floating-point representation. The best way to answer this question is to convert this real number into a 6-bit IEEE-like floating-point representation and clearly indicate why it would overflow.

k. How close is the value of the `frac` of the largest normalized number to 1? In other words, how close is **M** to 2? Yet another way of phrasing this question is to ask: what is the value of ε (epsilon) in this equation 1 <= **M** < 2 - ε? Express your answer as a fractional decimal number (i.e., a real number).