Assignment 1

COMP 2401

Date: September 17, 2018

Due: on September 30, 2018 by 23:55 (see below regarding Part I)

Submission and Due time:

Part I in the assignment box on the 4th floor. Part I can be submitted until 16:00 on October 1, 2018

Part II: Electronic submission on culearn. Part II is due on September 30, 2018 by 23:55

Do Not post the assignment or the assignment and solutions on any website

Objectives:

a. Understanding number representation of integers and floats

b. Basic experience with writing code and submitting a coding assignment.

Submission

Submission must be in the assignment box by the due date and time.

Grading:

- Assignment is graded out of 100 (note that you can get more than 100)
- Part I maximum grade is 72
- Part II maximum grade is 30

Part I - Numbers Representation (72 pts)

- 9 points per question (1-8). Question 9 is a bonus question. Note that the total includes 10 discretionary points for submitting clean and readable assignment. It is up to the TA to deduct points for improper submission.
- Submission of Part I is in the course's assignment box

Show your work.

1) How many different values can be represented using 9 digits in Justify your answer

a) Binary Systems

b) Hexadecimal Systems

c) Octal Systems -

2) Find the decimal equivalent of the following:
(a) $(245)_8$

(b) $(FAC1)_{16}$

(c) $(312)_4$

(d) $(10110110)_2$ - the number is one byte long. Show a solution when the number is interpreted as a 2's complement number and when it is interpreted as an unsigned number

(e) $(00010111)_2$ - number is one byte long

3) Assume that there are 6-bits that are available to store a binary number. Specify the range of numbers that can be represented by the following number systems:

a) Unsigned Integers (i.e., only positive numbers) -

b) 2's Complement

c) 1's Complement

4) Perform the operation on the following 8-bit binary numbers. The numbers are represented in 2's-complement notation. Show the work using base 2.

a) 01011101 + 10101001

b) 10110111 - 11001011

c) 00000111 * 00000101

5) Convert the following decimal numbers into 1's complement, and 2's complement representations. Assume that 8 bits are used to store the numbers.

a) 72

b) 0

c) -128

d) -5

6) Find the decimal equivalent of the following binary numbers (1 byte long) assuming that each number is expressed in:
(1) unsigned integer (2) 1's complement (3) 2's complement (4) Excess 127

(a) 01011000 (b) 10111001

7) Convert the following decimal numbers to normalized floating point representation (e.g., 0.5 should be 1.0×2^{-1}) Show your work

(a) 3.375 (b) -1.25

8) Assuming that you are required to make an 8 bit floating point representation, where bit 7 is a sign bit, bits 4, 5, and 6 are the 3 bit exponent using excess-3 notation and bits 0, 1, 2, 3 represent a 4 bit mantissa using normalized notation.

a) How would the number 2.25 be stored in the byte?

b) What is the number that its binary representation is 11101010?

9) (Bonus question 10 points) Assuming that you are required to make an 8 bit floating point representation, with a 3 bit exponent using excess-3 notation and a 4 bit mantissa using normalized notation. How many numbers can you represent between 2.5 and 3(including 2.5 and 3).

Part II – programming (30 pts)

1 Code a Convert function

Write a conversion function that convert an integer number from base 10 to a number in another base. The function should print out the number in its new representation. Here you can extract the files in a01.tar. The files are convert.c, convert.h and main.c

Output examples
If the function is called with number 47 and the base 8 (e.g., convert(47, 10)) then the function should print

```
47
```

If the function is called with number 47 and the base 4 (e.g., convert(47, 8)) then the function should print

```
57
```

If the function is called with number 47 and the base 8 (e.g., convert(47, 2)) then the function should print

```
101111
```

Function prototype is **int convert(int num, int base);** where

input:
num - the number in decimal
base - the base to which num has to be converted to

output:
none
*
*/

The algorithm is the same one that was discussed in class for base=2.
Note that the algorithm produces the least significant digit first (namely the units) followed by the "tens" digit, etc. Thus, in order to accomplish the task you will need to write a recursive function where the print statement comes at the end

Pseudo Code

If number is 0 then print a 0
Otherwise
    Store the remainder ← num modulo base;

    modify num ← num / base which is the quotient of num and base

    recurse if num is not 0

    print the remainder

Here is an example of running the algorithm above when converting the number 27 in base 10 to its representation in base 5.

1. remainder = 27 mod 5 (remainder = 2) // remainder ← num modulo base;
2. num = 27 /5 (num = 5) // num ← num / base;
3. num is not 0 and therefore recurse
    a. remainder = 5 mod 5 (remainder = 0) // 1remainder ← num modulo base;
    b. num = 5 /5 (num = 1)
    c. num is not 0 and therefore recurse
        i. remainder = 1 mod 5 (remainder = 1)
        ii. num = 1 / 5 (num = 0)
        iii. num is 0 and therefore
        iv. print the remainder (print 1)
    d. print the remainder (print 0)
4. print the remainder (print 2)

The printed result is
102

Namely 27 in base 10 is 102 in base 5

2 Test your code using the test code in the file convert.c

Test your code. An example of test code is given in the function main() in the file convert.c. In order to compile and test your code you will have to use

gcc main.c convert.c

to produce an output file that is called convert use

gcc -o convert main.c convert.c

Fix any errors that may occur

3 submission

1. Create a readme file("README.txt") that includes:

   a. a preamble (program author, purpose, list of source files)
   b. exact compilation command
   c. launching and operating instructions

2. Create a tar file called "a1.tar" that has your convert.c, convert.h, main.c and README.txt files in it. There should be no directories in the tar file.

3. Submit your a1.tar file in cuLearn.