**ECE 1111: Engineering Computation I**

**Homework No. 7: Bitwise Operators, Masking and Character String Conversions**

**Goal:** Demonstrate that you understand how to do bit-level operations and character string conversions in C. *DO NOT COPY CODE FROM THE INTERNET THAT YOU DON’T UNDERSTAND.* Write the code yourself.

**Description:** There are three tasks in this homework assignment:

1. Write a program that reads any file of text containing one character per line and prints the corresponding bit values for each character. You can look up the decimal values of the characters using the ASCII chart we covered previously to debug your code. You can access the individual bit values using bit shifting and/or masking.

Demonstrate that your code works by processing this file as a test case:

nedc\_000\_[1]: more v.txt

a

b

c

0

1

Your output should look like this:

a => 01100001 (97)

b => 01100010 (98)

...

1. Write a program that demonstrates how to mask the above data file with the bit pattern "0F". The program should accept a filename and a mask value from the command line. It should open the file, read it line by line, and process the first character value in each line. Demonstrate logical AND, OR, and XOR operations. Do this by printing the following (as an example):

ece-000\_[1]: p02.exe v.txt “0F”

mask = 0F

Input: a bits: 01100001 and: 00000001 or: 01101111 xor: 01101110

Input: b …

Input: c …

1. Write a program that reads a file line by line and ignores any line for which the first non-whitespace character is a “#”. Create a file containing ASCII text like this example to test your program:

nedc\_000\_[1]: more v.txt

# an example file

#

this is text

27

another line of text

-27

yet another line of text

27.272727

yet another line of text

-27.272727

yet another line of text

3.3e-01

yet another line of text

-3.3e-01

If the line contains a number, print the value of the number as a floating-point value using a format of “%10.4f”. Otherwise print the line as a character string. Be sure not to introduce an extra newline character – your output should be single-spaced.

Create three directories: *p01*, *p02* and *p03*. Within these directories, submit three programs: *p01.cc*, *p02.cc* and p03.cc. They should compile and link using a make file.