Part 1

The first part of this assignment is problems that you will **NOT** turn in for a grade. Calculate answers by hand and then write C programs that verify your answer (and vice versa). Make sure you understand both methods.

3.36 Suppose a, b and c are integer variables that have been assigned the values a = 8, b = 3 and c = –5. Determine the value of each of the following arithmetic expressions.

(*a*) `a + b + c`
(*b*) `2 * b + 3 * (a - c)`
(*c*) `a / b`
(*d*) `a % b`
(*e*) `a / c`
(*f*) `a % c`
(*g*) `a * b / c`
(*h*) `a * (b / c)`
(*i*) `(a * c) % b`
(*j*) `a * (c % b)`

3.38 Suppose c1, c2 and c3 are character-type variables that have been assigned the characters E, 5 and ?, respectively. Determine the numerical value of the following expressions, based upon the ASCII character set (see Table 2-1).

(*a*) `c1`
(*b*) `c1 - c2 + c3`
(*c*) `c2 - 2`
(*d*) `c2 - '2'`
(*e*) `c3 + '#'`
(*f*) `c1 % c3`
(*g*) `'2' + '2'`
(*h*) `(c1 / c2) * c3`
(*i*) `3 * c2`
(*j*) `'3' * c2`

3.42 Each of the following expressions involves the use of a library function. Identify the purpose of each expression. (See Appendix H for an extensive list of library functions.)

(*a*) `abs(i - 2 * j)`
(*b*) `fabs(x + y)`
(*c*) `isprint(c)`
(*d*) `isdigit(c)`
(*e*) `toupper(d)`
(*f*) `ceil(x)`
(*g*) `floor(x + y)`
(*h*) `islower(c)`
(*i*) `isupper(j)`
(*j*) `exp(x)`
(*k*) `log(x)`
(*l*) `sqrt(x*x + y*y)`
(*m*) `isalnum(10 * j)`
(*n*) `isalpha(10 * j)`
(*o*) `isascii(10 * j)`
(*p*) `toascii(10 * j)`
(*q*) `fmod(x, y)`
(*r*) `tolower(65)`
(*s*) `pow(x - y, 3.0)`
(*t*) `sin(x - y)`
(*u*) `strlen("hello\0")`
(*v*) `strpos("hello\0", 'e')`

Perform a google search of the function to discover what header file to include in the code needed in order to use the function

4.50 A C program contains the following statements:

```
#include <stdio.h>

char a, b, c;
```

(*a*) Write appropriate `getchar` statements that will allow values for a, b and c to be entered into the computer.

(*b*) Write appropriate `putchar` statements that will allow the current values of a, b and c to be written out of the computer (i.e., to be displayed).

4.51 Solve Prob. 4.50 using a single `scanf` function and a single `printf` function rather than the `getchar` and `putchar` statements. Compare your answer with the solution to Prob. 4.50.

13.39 Suppose that a is an unsigned integer whose value is (hexadecimal) `0xa2c3`. Write the corresponding bit pattern for this value. Then evaluate each of the following bitwise expressions, first showing the resulting bit pattern and then the equivalent hexadecimal value. Utilize the original value of a in each expression. Assume that a is stored in a 16-bit word.

(*a*) `~a`	(*h*) `a >> 3`	(*o*) `a & ~(0x3f06 << 8)`
(*b*) `a & 0x3f06`	(*i*) `a << 5`	(*p*) `a ^ ~0x3f06 << 8`
(*c*) `a ^ 0x3f06`	(*j*) `a & -a`	(*q*) `(a ^ ~0x3f06) << 8`
(*d*) `a | 0x3f06`	(*k*) `a ^ ~a`	(*r*) `a ^ ~(0x3f06 << 8)`
(*e*) `a & ~0x3f06`	(*l*) `a | -a`	(*s*) `a | ~0x3f06 << 8`
(*f*) `a ^ ~0x3f06`	(*m*) `a & ~0x3f06 << 8`	(*t*) `(a | ~0x3f06) << 8`
(*g*) `a|~0x3f06`	(*n*) `(a & ~0x3f06) << 8`	(*u*) `a | ~(0x3f06 << 8)`

- It may be necessary to use short int as a data type to get a 16-bit data field

Part 2

The second part of this homework is a program that should be written per the description below and turned in to the Canvas assignment for Homework #3. Turn in just a single C source code file for the assignment. The program will be tested with gcc under Cygwin.

Write a program that takes two **command line arguments** at the time the program is executed. You may assume the user enters only decimal numeric characters. The input must be **fully qualified,** and the user should be notified of any value out of range for a **23-bit unsigned integer**. The first argument is to be considered a data field. This data field is to be is operated upon by a mask defined by the second argument. The program should display a menu that allows the user to select different bit-wise operations to be performed on the data by the mask. The required operations are: Set, Clear and Toggle. **Both data and mask should be displayed in binary format**. And they must be displayed such that one is above the other so that they may be visually compared bit by bit. If data and mask are both less than 256, then the binary display should only show 8 bits. If both values are less the 65536, then the binary display should only show 16 bits. The menu must also allow the user to re-enter a value for data and re-enter a value for the mask. Use scanf() to read data from the user at runtime and overwrite the original values provided at execution time. All user input must be completely qualified to meet the requirements above (up to 23-bit unsigned integer values).

Printing binary must be done with shifting and bitwise operations. Do not use arrays. No multiplication or division is needed to produce binary output to the screen.