CSC 230 Assignment 2
Fall 2017

This assignment is a C programming assignment needs to be tested on the lab's computers, using gcc. When your program is complete and tested, including appropriate documentation, submit it on Connex under the Assignments, Assignment 2 link.

Submit a C file called bitFact.c on conneX assignments by Monday, September 25, 2017 well before 5 am.

This is a C programming assignment. It can be developed on any system of your choice, however, it will receive a grade of 0 if it does not work on the computers in ECS 249 using the compiler command line "gcc -Wall -std=gnu99 -o bitFact.o bitFact.c" and then running the executable.

Learning Objectives

Upon successful completion of this assignment, you should be able to write C programs that:

- ✓ Use various C data types
- ✓ Create, store data in and read from C arrays.
- ✓ Write and call C functions that receive input parameters and return values.
- ✓ Use the C bitwise and shift operators.

The problem to be solved by this program is to input an integer, calculate the factorial of that integer and output the result in decimal and binary. There is a qualifier, the individual bits of the binary form must be each stored into a separate element of an array.

A sample execution of a program that follows the specifications is:

```
FACTORIAL & BIT TESTER

Input a positive integer ==> 7

  7     Factorial =  5040 or 0x13b0 or 0b0001001110110000

Do another (y/n)? y

Input a positive integer ==> 12

  12    Factorial = 64512 or 0xfc00 or 0b1111110000000000
```

Notice that the hexadecimal output is optional and not described in the specifications below.

Since the integers used throughout this assignment are unsigned short, which are 16 bits wide, it will be convenient to have the following constant defined using the pre-processor:

```
# define SIZE_INT 16
```

Here are the specifications of the functions to be used:

- Write a function that accepts an array of **`unsigned char`** sized integers describing an integer in binary format (each item in the array corresponds to a single bit of the integer) and prints each element of the array as a decimal number. The function prototype (or signature) is:

  ```
  void printBitArray(unsigned char theBits[SIZE_INT]);
  ```

- Write a function accepts an **`unsigned short`** sized integer value and an array of **`unsigned char`** sized integer values. The function places each of the 16 bits of unsigned short integer into different array elements, in the least significant bit location. If, for example, the input integer was 25 (= 0x0019 = 0b0000000000011001) then the array would need to contain:

0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1

  The function prototype (or signature) is:

  ```
  void toBits(unsigned short value,unsigned char inBits[SIZE_INT]);
  ```

- Write a *recursive* function that accepts an **`unsigned short`** sized integer parameter, calculates and returns the factorial of that value. The return value should also be an `unsigned short` integer. The function prototype (or signature) is:

  ```
  unsigned short factorial(unsigned short num);
  ```

- Write a **`main`** function that, inputs an integer number (of size **`unsigned short`**), calculates the factorial of that number, uses **`toBits`** to place the 16 bits of the result into 16 separate locations of a byte sized integer array, and then uses **`printBits`** to output the contents of that array. Finally, the program should give the user the option of re-running with different input.

For C programs written in CSc 230, we will choose to follow a previously defined style guideline: https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html. Review the guideline and ensure your code follows the recommendations of the authors.

Observe that the development of the solution for this problem could occur on your own system, but must be tested on the lab computers before final submission.