ECE199JL: Introduction to Computer Engineering **Fall 2012**
Homework 4 **Due: at start of lecture on Friday 21 September**

Introductory C Programming

Please read and note the guidelines for preparing and submitting your homework solutions as posted on the course web board; you are responsible for knowing and following the guidelines if you wish to receive credit for your work.

1. Program Modification and Enhancement
Download or copy the example program "`factorial.c`" that was given in the lecture. (Available at `http://courses.engr.illinois.edu/ece199/hw/factorial.c`, or copy from the course files on the Engineering WorkStation servers (`cp -r /class/ece199/hw4 .` and `cd hw4`) and modify the files therein.) Make sure you select a formatting style you like and put your name on any modified versions you create; one way you can mark your own work and satisfy the Creative Commons Attribution License requirements is to include a line like

```
/* This code is modified from "factorial.c" by Douglas L. Jones, as downloaded from
http://courses.engr.illinois.edu/ece199/hw/factorial.c on September XX, 2012 */
```

in your program's header.

The example `factorial.c` code gives erroneous answers for integer inputs that are negative-valued, and for values for which the answer exceeds $2^{31} - 1$. Your mission is to extend this program so that instead it prints the warning messages "`The factorial operation is undefined for negative integers`" or "`The factorial value exceeds the supported numerical range`", respectively, instead of an erroneous output value. (This kind of "range" checking is essential for codes that others might use, and even for codes that you might use again later after forgetting the details and limitations of your implementation!)

A. Create a flowgraph for your new, enhanced version of this program. (Turn it in as part of your homework solution.)

B. Write, compile, run, test, and debug your program to confirm that it provides the correct output for integer inputs within 32-bit range. Recall and review from Section 3.4 of Lab 0 how to compile and run C programs. (We recommend that you compile with all warnings enabled: `gcc -Wall -o factorial factorial.c`; the warnings often detect and alert you to program bugs.) Submit a printout of your code as your solution.

2. Program Analyis and Completion
In mathematics, Fibonacci series are the numbers in the following integer sequence. 0,1,1,2,3,5,8,13,21,34,55,89,... By definition, the first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.

Finish the following code to print n terms of the Fibonacci series.

```
/* fibonacci.c */
int main()
{
    /*Declare variables*/
    int num, counter, curr, prev1 = 0, prev2 = 1;

    /*Get user-defined number of terms*/
    printf("Please enter the number of terms:\n");
    scanf("%d", &num);

    /*Printing starts*/
    printf("%d-term Fibonacci series:\n",num);
```

```
   for (counter=0; counter<__________; counter++) {
       if (counter<=1) {
    /*Define current value of the sequence*/
           curr=__________;
       }
       else {

/*Compute current value of the sequence*/
           curr=_________+__________;
    /*Reinitialize two previous terms*/
           prev1=__________;
           prev2=__________;
       }
    /*Print Fibonacci series*/
       printf("%d\n",curr);
   }
   return 0;
}
```

3. Your First(?) C Program

Develop a C program to determine whether or not a positive integer $d < 2^{31}$ is a power of two.

A. Come up with your high-level strategy for solving this problem. (There are many viable approaches!) As your answer, describe your strategy in no more than two sentences. (For examples: "Count the number of bits ..." or "Compare the input with each ...")

B. Create a detailed flowgraph implementing the strategy you describe in Part A, along with a list of the data or variables needed. Submit this flowgraph as part of your homework solution.

C. Translate your flowgraph into a C program that correctly implements your algorithm. Format and comment your program so that whoever grades it can easily understand and evaluate it. Compile, debug, and adequately test your program to make sure it works correctly. Turn in a printout of your program as part of your homework solution.