# STAT 243 ASSIGNMENT 1 FALL, 2010

*due October 4, 2010*

## 1. Function Prototypes

Consider the following function, which adds two double precision floating point numbers:

```
double sum(double x, double y)
{
   return(x + y);
}
```

For this assignment, create a file containing this function definition, and a second file with a program which calls the `sum` function using two **integer** arguments, and which prints the value returned by `sum`. Compile and run the program twice, first declaring sum as

```
double sum();
```

and then using a function prototype, like

```
double sum(double,double);
```

Does the value printed by the program when you use the first declaration agree with the value when you use the function prototype? Explain why or why not.

## 2. Determining Machine Constants

Write a program (or programs) to determine the following:

- **Largest Integer** What is the largest integer (both long and short) that can be handled in the computer? What happens when an integer becomes larger than this due to a computation?
- **Largest Unsigned Integer** What is the largest unsigned integer (both long and short) that can be handled in the computer? What happens when an unsigned integer becomes larger than this due to a computation?
- **Largest Floating Point Number** What is the largest floating point number (both single and double precision) which can be handled in the computer? What happens when a number becomes larger than this due to a computation?
- **Machine Epsilon** One important constant on the computer is the smallest number such that, when it is added to 1, the result is greater than 1. What is the value of machine epsilon?

The simplest way to determine these constants is to write a program which starts with the number 1, and then multiplies or divides it by 2, printing out the result at each stage. These results can be examined, and it is usually pretty clear what the value of the constant is. A better solution is to write a program which can detect when the constant has been found, and to break out of the loop and print the answer at that time. The first solution is acceptable, but try to aim for the second method.

*Note:* Most of the values described above can be found in a header file in the standard include directory `/usr/include`. If you can find the appropriate header file, it may be of interest to compare the results in that file with the results of your program. (Please note that finding the header file does not constitute completing the assignment.)

## 3. Probability Tables for the $\chi^2$ Distribution

Write a program that will prompt a user for degrees of freedom and probability level, and which will then print the critical $\chi^2$ value. You will need to use a subroutine from a library such as `dcdflib` to calculate the critical value; look at `/app/dcdflib/doc/dcdflib.fdoc` for information on how to call these routines. You should have some provision in your program to allow the user to make multiple requests, and then terminate the program in a "graceful" way.

*Note:* The dcdflib routines are written in FORTRAN and you'll write your main routine in C so you'll need to learn how to call FORTRAN routines from C. Type `help fortran_from_c` for more information.