Computing and Statistical Data Analysis
Problem sheet 1
Due Monday, 18 October, 2009

Coding style counts. Remember to use meaningful names for variables and indent properly; I suggest two spaces per indentation. Please do not go past 80 characters per line or else the code becomes difficult to read when printed (in C++ you can always break a line wherever you like).

Problem 1 is optional. Do those parts that involve something that you haven't encountered previously. (Nothing to turn in for #1.)

1(a) Log in to your unix account and try out all of the commands listed in the notes from lecture 1. (Nothing to turn in here other than to report any unexpected behaviour.)

(b) Using the man pages, find out what the command grep does, and give an example of its use. (You may learn more by googling for grep.)

(c) Create a subdirectory called HelloWorld. There create (using, e.g., xemacs) a file called HelloWorld.cc, in which you enter the code from lecture 1.

(d) Compile and link the code using the g++ command as indicated in the lecture. Ensure that the program compiles and runs correctly.

(e) Enter the compile command into a shell script called `build.sh` and build the code using this.

2: To do this exercise we need to use a loop, which we will introduce in week 2. To get a start on this without using a loop, you can begin by using Stirling's approximation for all n (see below). Write a small C++ program that does the following:

- The program should prompt the user for a non-negative integer, whose value is assigned to a variable n.
- If the value of n is 50 or less, compute $n!$ by "brute force" and write the result to the monitor using `cout`. Think about whether it would be better to store the values as integer or real (e.g., double).
- If the value of n is greater than 50, compute $n!$ using Stirling's (improved) approximation:

$$\ln n! \approx n \ln n - n + \tfrac{1}{2} \ln(2\pi n) \; .$$

  You will need to put `#include <cmath>` at the beginning of your program in order to use the logarithm and exponential functions, `log` and `exp`.
- Your program should check whether the input value is a valid argument for the factorial function (i.e., non-negative integer) and it should take appropriate action if this is not the case. Think what to do if, say, a value of 3.7 is entered.
- Determine roughly the maximum value of n for which your program is valid. (Try entering a very large value of n, say, 1000 – what happens? Try then scaling this number up or down by factors of two until you determine the maximum value.)

G. Cowan
3 October, 2010