# Math 683 Assignment 1

| | |
|---|---|
| **Professor:** | *Richard Hall* |
| **Instructions:** | *Please use file names and identifiers exactly as requested. For example, for this assignment I should like precisely 4 files called intfun.cpp, intfun.exe, complex.cpp and cxtest.exe on a floppy disk, along with annotated print outs of the* `.cpp` *files. You are welcome to use 'our'* `w.h` *'include' file in your code.* |
| **Due Date:** | *26 Sept 2000* |

(1.1) Design the functions `int roll(int i, int n, int d = 1)`, `int gcd(int m, int n)` and `int fibnr(int n)` and put them at the top of the file `intfun.cpp`, which also includes a small program to test the three functions. The function roll yields the integer obtained by starting from $i$ in $\{1, 2, 3, \ldots, i, \ldots n\}$ and stepping $d$ steps in cyclic fashion (if reached, the next step after $n$ is $1$). Thus $\text{roll}(3, 9, 12) = 6$. The gcd function yields the greatest common divisor: this should be designed as a recursive function. The function fibnr is a *non*-recursive Fibonacci function yielding values $\{1, 1, 2, 3, 5, 8, \ldots\}$.

(1.1a) When you are satisfied with `roll`, it can be added to the `.cpp` section of `w0.h`. Similarly, the definitions of the function `ftab(fptype1, ...)` and its overload `ftab(fun *, ...)` can also be added. The now 'complete' revised file can be renamed `w.h`. I do *not* need to see this file: it is for your own use.

(1.2) Complete the class `Complex` introduced in the lectures and discussed in various places in Capper: please do *not* use the identifier `complex` since it may already be in use in your environment. The header file `complex.h` including the declarations is provided, so is the file `cxtest.cpp` to be compiled : you are required now to supply the definitions in a file `complex.cpp`, so that the given `cxtest.cpp` will indeed compile to `cxtest.exe`. Read Capper carefully on the topic of the Complex class. Include also the functions `norm2` = $|z|^2$ , `norm` = $|z|$ and `Complex power(const Complex &z, double r)`, which yields the power $z^r$, where $r$ is real. Add also `Complex powi(const Complex &z, int n)` which is defined recursively and computes the non-negative integral power $z^n$. Notice that we do *not* need a special constructor with no parameters for arrays because the first constructor has zero default values: thus `Complex c;` means c = (0,0), and `Complex c[100];` constructs an array of the same. The output functions `p` and `pl` should 'print' a complex number to the console in some 'sensible' (perhaps attractive) fashion. You may need to extend this useful class for later assignments.