1. User accounts have been created under the names p420u1xx with xx ranging from 00 to 19 and p580u1xx with xx ranging from 00 to 12. Please claim one of these userids by adding your name to the list at the front of the class.

2. Your initial password has been set to phys_420 or phys_580, accordingly. Your first order of business after successfully logging in is to change this password to one of your own choosing. Open a terminal window from the Applications ▷ Accessories ▷ Terminal menu located in the top-left corner of the screen and run the command yppasswd.

3. Work through some of the examples and exercises from the Linux/BASH/C++ tutorial posted on the class website, http://www.ualberta.ca/~kbeach/phys420_580.html. If there is anything you do not understand, please ask questions.

4. A Lissajous figure refers to a planar trajectory that is harmonic in two orthogonal directions. This is something you might have seen traced out on an oscilloscope. You can find more details here: http://en.wikipedia.org/wiki/Lissajous

Write a C++ program that computes the quantities

\[ x(t) = A \cos(at + \delta) \]
\[ y(t) = B \cos(bt) \]

at 100N equally-spaced points in the range \( 0 < t < 2\pi N \times \max(1/a, 1/b) \) and outputs the results in three-column format \( t, x(t), y(t) \) to the standard output stream. Make it so that your program requires six command line arguments. The first five should be interpreted as floating-point numbers (with the atof function) and used to set the values of \( A, B, a, b, \delta \). The sixth should be interpreted as an integer (with atoi) and assigned to \( N \). The program output can then be written to a file via redirection (>) and viewed with gnuplot.

\[ g++ -o lissajous lissajous.cpp \]
\[ ./lissajous 2.6 1 3 2 0.5 2 > curve1.dat \]
\[ ./lissajous 1 1 1.1 1.2 0 35 > curve2.dat \]
\[ gnuplot \]
\[ > plot "curve1.dat" using 2:3 with lines \]
\[ > plot "curve2.dat" using 1:($2+$3) with lines, 2*cos(0.05*x), -2*cos(0.05*x) \]
\[ > quit \]

(a) Convince yourself that a Lissajous figure is closed iff \( a/b \) is a rational number.
(b) How does the ratio \( a/b \) control the shape of the curve?
(c) In the case \( a = b \), how does the phase shift \( \delta \) effect the curve?
(d) Investigate the beats produced when the two sinusoidal components—with equal amplitudes and slightly different frequencies—are superimposed. In other words, plot \( z(t) = x(t) + y(t) \) versus \( t \) for \( A = B \) and \( |a - b| \ll 1 \). The result is a product of a slowly varying envelope function and a rapidly varying beat function:

\[ \cos(\alpha t) + \cos(\beta t) = 2 \cos\left[\frac{1}{2}(\alpha + \beta)t\right] \cos\left[\frac{1}{2}(\alpha - \beta)t\right] \]

5. The Mandelbrot set consists of the bounded orbits of the complex-valued recurrence relation

\[ z_{n+1} = z_n^2 + c, \quad z_0 = c \equiv x + iy \]
The set is typically visualized as a plot in the $x$-$y$ plane, with each point corresponding to an unbounded orbit coloured according to its rate of escape. For more information, take a look at this article: http://en.wikipedia.org/wiki/Mandelbrot_set

Write a C++ program that implements the following algorithm. Scan over a grid of $c$ values such that its real and imaginary parts range over $x \in [-2, 1]$ and $y \in [-1, 1]$. At each point, run the recurrence relation until $|z_n| > R$ or $n > N$. I suggest the values $R = 3$ and $N = 500$. Store the escape count $n$ for that point in an array. Output the array as a rectangular table of values to stdout. Plot the set using gnuplot:

```
$ g++ -o mandelbrot mandelbrot.cpp
$ ./mandelbrot > mandelbrot.dat
$ gnuplot
gnuplot> set pm3d map
gnuplot> splot "mandelbrot.dat" matrix
```

6. Before you leave, make sure to close out of your session from the System ➤ Logout menu.