Introduction to *Macaulay 2*

When learning a new computer program the most important thing to know is how to use the help function. It also helps to have a healthy sense of persistence and curiosity.

Starting M2 Open emacs. Then use F12. If M2 is properly set up on your machine this should work. If you are using a Windows machine you will need to start up cygwin first. Emacs can be used from the menus without much trouble, but there are many key-strokes that can make things easier/quicker for you. Here are a few of my favorites.

- C-x C-s save
- C-x C-f open a new file
- C-x 3 split the screen vertically.
- C-x b change buffers, by name.

It is also helpful to know naming conventions for any computer algebra system you use. Macaulay2 has a naming convention — objects (rings, ideals, modules, etc.) are capitalized and the function to create those objects are lowercase. Functions are lowercase, unless they consist of two words, like primary decomposition and then the second word is capitalized, so `primaryDecomposition` is the function that computes a primary decomposition.

Getting Help

- To get a help browser on a **mac or unix** based machine, just type viewHelp into *Macaulay 2*. For a **windows** machine, this may also work. However, since *Macaulay 2* is still cygwin based, you are likely to need to do the following.

  Open `My Computer`. Then the `cygwin` folder. Then `usr`, then `local`, then `Macaulay2`, then `share`, then `doc`, then `Macaulay2` and then `Macaulay2` again. Here there should be a folder called html. The address box should read

  `C:\cygwin\usr\local\Macaulay2\share\doc\Macaulay2\Macaulay2\`

  Sadly you cannot just type in the address.
  You can copy this to the desktop (make it easier to find later - but you lose a picture and I don't guarantee all links work). Either here or from the desktop (if you copy it), open the file, scroll to the very end and double click on `index.html`. This will open the current documentation for you.

- To get help on a particular function type `help functionName` and hit return. This is particularly useful if you know or have a guess at the function name, but don't know the syntax. For example, below we will suggest some functions, like res, and primaryDecomposition, but I don't indicate the syntax and a great way to find this quickly is by typing `help res`.

- There are two other good resources as your skills advance.
  - There is a *Macaulay 2* Google group. Consider subscribing and searching for answers there. If you still can't figure out the answer (after using the documentation and searching the group) consider posting your question. Someone in the community usually answers very quickly.
  - There is a series of wiki pages developed by the approximately annual *Macaulay 2* workshops (consider attending one in the future, Summer 2011 IMA and Summer 2012 at Wake Forest).

    `http://wiki.macaulay2.com/Macaulay2/index.php?title=Main_Page`

    Every workshop has a page labeled "How To" or "Information from How To," etc. and starting with CC2010 there is "Five things I wish I knew." Both of these are the pages that you might find useful once you start writing packages.

Basic Functions

- To make a ring, the syntax is much like you would write it on a piece of paper, like $k[x, y, z]$ or $k[x_1, ..., x_n]$. You have to actually specify a base field (although $\mathbb{Z}$ is also allowed) and you can change the monomial ordering and degrees of the variables using a couple of optional arguments. Here are examples of what you might type into Macaulay2 - the default monomial order is graded reverse lexicographic.

```
R = QQ[x,y,z, MonomialOrder => Lex]
R = ZZ/32003[a..e, Degrees => {{1,0},{1,0}, {1,1}, {1,1}, {1,2}}]
R = ZZ/5[x_1..x_10, MonomialOrder => ProductOrder{2,8}]
```

  There are other orders available and degrees can be done to give variables higher degrees as well as multi-degrees. Check out viewHelp and look in the index under monomial orderings and see what you find there.

- Making ideals is, again, like you might write it on a piece of paper, except that we have to tell it what we are making.

```
I = ideal(x^2, y^3-z, y*z-4)
J = ideal"a2-b3, 4aed+5ac2d"
```

  Note that there are two ways to enter an ideal. I usually use the first but you might find you like one better than the other.

- Two other functions that are handy right away are `matrix` and `map`

```
M = matrix{{x,y,z}, {x^2, y^2, z^2}}

R = QQ[x,y,z]
S = QQ[t]
phi = map(S,R, {t^2, t^5, t^9})
kernel phi
```

- Beyond this there are functions for most things you might want to do; resolutions, dimension, codimension, kernel, image, primary decomposition, Ext, Tor, and the list goes on. The best way to use these functions once you can make things like rings and ideals, is to use the help features.
  - Pick an ideal for which the primary decomposition is easy to compute by hand, like (x^2, xy) or $(x^2 - y, z^2)$. Write it down, then check it using *Macaulay 2*. **Key Note**: *Macaulay 2* has an object MonomialIdeal. If you set up the first ideal as a monomial ideal (think about the conventions for commands and what to do should seem natural to you) then *Macaulay 2* automatically uses the much more efficient methods for primary decompositions of monomial ideals.
  - Pick an ideal for which the integral closure of R/I is easy to compute by hand, for example (x^2, y^2) or $(x - y)$ in $k[x, y]$ and then use *Macaulay 2* to do the computation.
  - Pick an ideal for which the resolution is easy to compute, like any regular sequence, and see that you indeed get the Koszul resolution.

- Finally, lets take a look at *Macaulay 2* and modules. The help page found under "modules" (not module) is a good reference for this.
  - Free modules: Just type R^{degrees} where degrees here is a list of the appropriate degrees for the generators. This can be a list of lists for multi-graded modules.

- Computing a resolution is a great way to generate lots of modules. Many of these are most easily accessed using image, kernal, or cokernal of the corresponding map — in fact these are great for any map, regardless of how you build the map. Of course matrices will be the standard way to build maps. To access the maps in a resolution use $r.dd_i$ where r is the name you gave the resolution and i indicated which map you want.
- For submodules and quotients, we can do what seems natural. For example, build an ideal I in a ring R and a free module $M = R^3$, for example. Then try $N = I * M$. To get at the generators do N_i where i is a number between 0 and the total number of expected generators. For quotients, say M/N just type this exactly.
- What happens when we are considering modules that are not submodules or quotients of free modules? *Macaulay 2* represents these modules as "subquotient modules" e.g. given two maps with the same target $f : R^m \to R^n$ and $g : R^p \to R^n$ then these modules are of the form $(\text{image} f + \text{image} g)/(\text{image} g)$. On the page modules there is a link subquotient modules which is very good and has examples of how to access all the information you would want, in this case.

Of course, then there is homomorphisms of modules, computing chain complexes, Hilbert functions, and on and on. The modules page in the documentation is the place to start.

- Get creative, try to compute your favorite commutative algebra object. What happens? Ask for help if you need it!
- One last key command for this workshop. We will need some packages that are not preloaded, but are distributed with *Macaulay 2*. To access these packages and their documentation use

```
installPackage"Binomials"
```

to install the package Binomials which is good for computing primary decompositions of binomial ideals. Once you install the package, loadPackage is good enough, but installPackage still works. However, do not do loadPackage and then installPackage in the same session (errors result). You may also find GraphicalModels, Graphs, and several of the toric packages, or packages which provide links to other programs like normaliz and 4ti2 useful during the course of this workshop.