**Bonus material for Section 2**

Wildcards

The tutorial mentions the '\*' wildcard which matches any string. Now suppose you had five

hundred pictures in a directory:

> ls

img0.png img10.png img20.png ... img100.png

img1.png img11.png img21.png img101.png

img2.png img12.png img22.png

img3.png img13.png img23.png

img4.png img14.png img24.png ...

img5.png img15.png img25.png

img6.png img16.png img26.png

img7.png img17.png img27.png img497.png

img8.png img18.png img28.png img498.png

img9.png img19.png img29.png img499.png

and you decided that pictures 10 through 19 were no good. So you do:

> rm img1\*.png

What will be the result?

There's another wildcard character that should be used in this situation, the '?'.

Look at the man page for the bash shell:

> man bash

It's really long--you need to find the section on 'Pattern Matching'. When you're in man pages, you can hit '/' and then type a word or phrase to search for. Try it out.

Once it finds a match, you can hit 'n' to go to the next match (if there is one).

How are '?' and '\*' different? How could you use ‘?’ to delete pictures 10-19 from above example?

Some other ‘gotchas’:

Is there any way to get a file back if you've deleted it with 'rm'?

What happens if you 'mv' a file to a name that already exists?

What happens if you 'cp' a file to a name that already exists?