1. What text editors/IDEs have you used before Emacs? What ideas are similar between them and Emacs? What's different?
    a. Homework hint! Struggling to remember all the commands? Print this cheatsheet out (or have it handy on a screen nearby) as you do the lab.
    b. Don't forget to run `M-x open-dribble-file` for each editing session!

Answers will vary. If you weren't at discussion, take a few minutes to think about this.

2. The command `env` lists all the set environment variables in your shell. What are some environment variables you see besides `PATH`?

Answers will vary, but some that I see on SEASnet include:
HOSTNAME=lnxsrv10.seas.ucla.edu
TERM=xterm-256color
SHELL=/bin/bash
SSH_TTY=/dev/pts/14
USER=nathans
PATH=/usr/local/cs/bin:/usr/lib64/qt-3.3/bin:/u/cs/ugrad/nathans/perl5/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/usr/local/cs/bin:/u/cs/ugrad/nathans/bin
PWD=/u/cs/ugrad/nathans
HOME=/u/cs/ugrad/nathans

The main commonality of environment variables is that they contain configuration information about your system like what shell and terminal you're using, what your home directory is, etc. Environment variables are also very useful for configuring web applications, and you can keep your configuration separate from your code!

3. The "Unix Philosophy" is the idea that programs should be small and do one thing well.
    a. What are some examples of this that we've seen in class?
    b. Have we seen any programs that don't follow the Unix philosophy?
    c. Dan Luu argues that increasing complexity is inevitable and ultimately better for users, as it gives them easier ways to accomplish tasks. Rob Pike and Brian Kernighan disagree, using the -v option of cat as an example.
    d. Who do you agree with? Why?

Programs that follow the Unix philosophy: cat, tac, ls, cd, rm, diff, find, grep
Programs that don't follow the Unix philosophy: emacs
Answers can go either way on the argument.

4. Can you ssh into `lnxsrv06.seas.ucla.edu?`

Hopefully you can! Make sure you're connected to the UCLA VPN. Come to office hours if you're still having trouble

5. Run the command `man man`. What does it say?
    a. You can navigate the man page by using the arrow keys to move up/down a line

b. Space goes forward one page, ‘b’ goes back one page
c. ‘q’ quits
d. Homework hint! Programs you’ll want to man include:
    i. `which`
    ii. `wget`
    iii. `cp`
    iv. `ls`
    v. `chmod`
    vi. `find`
    vii. `locale`
e. Are man pages too long for you? There are also TL;DR pages.

Answer:
Man man should say something like: “man - format and display the on-line manual pages”. Take some time to read through the man pages for some of the commands!

Shell Scripting

Shell Scripting Hints:

a. Remember to always start your scripts with the line “#!/bin/bash”.
b. If you try to run a shell script but you get the error “Permission denied”, make sure that you have given that file execution permissions by running:
   `chmod +x <file-name>`
   BN: Don’t just give execution permissions to any file! Before you execute any script, always make sure that you know who wrote it and what it does!

6. Write a shell script named “hello.sh” that, when run, outputs “Hello, world!”
   Answer:
   ```
   echo "Hello, world!"
   ```

7. Write a shell script named “first.sh” that outputs whatever its first argument is (empty if no arguments are given).
   Answer:
   ```
   echo $1
   ```

8. The following shell script prints out the number of visible files in the current directory:

   count-files.sh:
   ```
   #!/bin/bash

   FILES=`ls`
   COUNT=0
   ```

```
for FILE in $FILES
do
  # echo "Found file: $FILE"
  let COUNT++
done

echo "There are $COUNT visible files in this directory."
```

Can you use the above script to write a new one named “list-files.sh” that behaves exactly as the `ls` command (invoked with no arguments), except it prints out files separated by spaces instead of tabs, and terminates its output with a newline?

Hint: By default, echo always adds a newline character to the end of its output. You’ll need to figure out how to suppress this functionality.

Answer:

```
#!/bin/bash

FILES=`ls`
for FILE in $FILES
do
  echo -n “$FILE ”
done

echo
```