(2) 1. Explain what would happen if the following did not give error messages:

```
$ cat x y > y
```

and with

```
$ cat x >> x
```

Think before you rush off to try them. Assume both input files are very large.

By itself give proper inputs $ cat x y > y would attempt to merge the contents of file x and y into y, and without error messages, the user could attempt to merge the contents of two incompatible files which would either be an impossible maneuver or create junk in the new file.

Similarly, $ cat x >> x tries to append the contents of file x to itself which could be problematic without error messages if given a file that would not support such an operation like an image.

(2) 2. Write a short shell script called "cx" that will execute the command "chmod +x" on every file given on its command line. What does it do?

It gives execute permission to the user for every file given.

(2) 3. Write a short shell script called "nf" to display the number of files in the current directory.

(4) 4. Write a shell script called "lss" that will list all the files in the current directory in decreasing order of the number of bytes in the file. It does not need to take any arguments. Eg.,

```
$ lss
-rwxrwxr--    1 wayne   faculty     385 Nov 29  1996 lss
   -rwxrwxr--    1 wayne   faculty      42 Mar  9  1990 nf
```

(4) 5. Write a shell script called "whoson" to display a sorted list of undergrad students logged in on the current machine. You should take a look at the command called "groups" to acquire some of the requisite information. Test it on odin.ics.uci.edu. eg.,

```
$ whoson
frank jane jim john laura
```

(2) 6. Write a shell script called "howmany" to display the number of undergrad students logged in on the current machine. Use whoson if you can. eg.,

$ howmany
5

(4) 7. Write a shell script called "valid" that determines if it's argument is a valid shell variable name, eg

$ valid foobar
yes
$ valid 12foobar
no

(HINT: define a regular expression, and enlist the aid of "grep".)

(4) 8. Write a shell script called "prargs" that prints out a numbered list of its arguments in the following format:

$ prargs a 'b c'
0: "prargs"
1: "a"
2: "b c"