Important: Before you leave, log out of the computer. Do this with `Ctrl-d` or `exit`.

If you don't finish the lab today, the scripts are due by 4:00 PM on Friday. They must all be turned in with turnin.

Part 1

1. Log in and cd to your cs246 directory.
2. Edit a shell script called globbing.sh and insert comments at the top with your name, the assignment, and a description like this:

   ```
   #!/bin/bash

   # Name: Your Name
   # Assignment: globbing
   # Purpose: Globbing Practice
   ```

   Save the file with `Ctrl-x Ctrl-s`.
3. You're going to need to use the shell while editing the script. Split the screen vertically with `Ctrl-x 3`. Then run a shell in the other window with `Meta-x shell`. On our keyboards, the Meta key is Alt.
4. You should now have an edit window for your script on the left and a shell window on the right. You can switch back and forth with `Ctrl-x o` (the o is for other window). Try this out.
5. Next move to the shell window and split it vertically using `Ctrl-x 2`. You are going to need a second shell window. Notice that anything you type in one shell window is also displayed in the other.
6. To get a second independent shell you need to rename the first one. Do that with `Meta-x rename-buffer`. It will ask you for a buffer name. Give it `discworld-shell`.
7. Now do `Meta-x shell` again. This will start a new shell in the other window.
8. Switch to the other window and rename that shell also. Call it `script-shell`.
9. In this window, use `cat globbing.sh` to make sure that you have saved your script and that you are in your cs246 directory.
10. Move back to the discworld-shell window and cd to the directory containing course files.

    ```
    cd /home/mathcs/courses/cs246
    ```

11. List the files in this directory with `ls -l` . This directory always contains the test scripts and data files for programs that you turn in with turnin. You should see a subdirectory called discworld.
12. Use cd to move into the discworld directory. Then list the files with ls. This directory contains several files whose names are characters in the Discworld series of book by Terry Pratchett. Each file contains the title of the books in which the character appears, but we won't use the contents today.
13. To practice globbing, you are going to use glob expressions to list the files in this directory that match various patterns. For each one, you will add three lines to the shell script in the other window.
    - Use echo to print the part number (actually letter) of the exercise part from the list below.
    - Use the ls command with the -C option to list the files for a particular pattern.
    - Use echo to print a blank line.
14. For each pattern, test the pattern in the terminal window first by simply typing the ls command. Then switch back to the edit window and add it to the script.
15. You should also test the script periodically by saving it with `Ctrl-x Ctrl-s` , switching to the script-shell window, and running it. The script should look like this (I've done the first two parts for you):

```
#!/bin/bash

# Name: Your Name
# Assignment: globbing
# Purpose: Globbing Practice

echo Part 1
ls -C r*
echo

echo Part 2
ls -C *r
echo

 ...
```

Here are the patterns:

1. Files whose names begin with "r".
2. Files whose names end with "r".
3. Files whose names begin with "c" and end with "t".
4. Files whose names contain "th".

5. Files whose names begin with “h” or “v”.
6. Files whose names begin with “d” and whose third letter is “a”.
7. Files whose names contain either “j”, “k”, or “z”.
8. Files whose names contain two vowels (“a”, “e”, “i”, “o”, or “u”.
9. Files whose names contain at least two ocurrences of the letter s.
10. Files whose names contain two consecutive vowels.
11. Files whose names end with a vowel.
12. Files whose names begin with a vowel but do not end with a vowel (warning: not vowel is not the same as consonant).
13. Files whose names contain either “gh” or “tu”.
14. Files whose names begin with either “ri” or “es”.
15. Files whose names contain an ”s” and an ”i”, in that order, but possibly with characters in between.
16. Files whose names have length 4.
17. Files whose names have length 9 or longer.
18. Files whose names have a character that is not a lowercase letter.

Part 2

Here is a script that takes a command line argument (call it n) and prints a square of characters of size n.

```
#!/bin/bash

# Name: Zaphod Beeblebrox
# Assignment: square
# Purpose: Print a square of characters. The character
#          is the first argument. The size is second.

for ((i = 0; i < $2; i++))
do
    for ((j = 0; j < $2; j++))
    do
        echo -n $1
    done
    echo
done
```

1. You won’t need the discworld window anymore so switch to it and make it go away with `Ctrl-x 0` .

2. In the emacs window, open the file square.sh: `Ctrl-x Ctrl-f square.sh`, enter the script, and save it.
3. Switch to the terminal window, make the script executable with `chmod +x square.sh` and test the script with `./square.sh`.
4. When you have the script working, make a new script called triangle1.sh by copying square.sh with `cp square.sh triangle1.sh`.
5. In the edit window, open triangle1.sh with `Ctrl-x Ctrl-f triangle1.sh` and modify it to print a triangle like this:

```
a
aa
aaa
aaaa
aaaaa
aaaaaa
```

   if the first argument is a and the second is 6. When you have the script working, test it in the terminal window.

6. Create a script triangle2.sh that prints a triangle of this shape:

```
ccccc
cccc
ccc
cc
c
```

7. Create a script triangle3.sh that prints a triangle of this shape:

```
ccccc
 cccc
  ccc
   cc
    c
```

8. Create a script triangle4.sh that prints a triangle of this shape:

```
    c
   cc
  ccc
 cccc
ccccc
```

9. Create a script diamond.sh that prints a diamond of this shape:

```
    x
   xxx
  xxxxx
 xxxxxxx
xxxxxxxxx
 xxxxxxx
  xxxxx
   xxx
    x
```

   If the second command line argument is n then the height and width will be $2n - 1$. In the example the argument is 5, so the height is 9.

Each script has to take the character to print and the size as arguments. You do not have to any error checking for these scripts. When you are finished with each script, turn it in with turnin.

You can run turnin from your script-shell window or you can exit emacs and then run it. Make sure that you log out before you leave.

If you are working in the lab outside of class, you can, of course, use a terminal with the window system activated. If you start emacs like this: `emacs -nw file` it will run as though the window system is off and you can practice the things we've worked on. If you log in from outside the lab with ssh, you will not have a window system, but you can use the techniques we've worked on to work efficiently within emacs.