CS 279 - Homework 2

Deadline

11:59 pm on **Friday, September 16**

Purpose

To practice using directory nicknames, to practice writing file-related commands, and to practice using shell local variables.

How to submit

You will complete **Problem 1** on the course Canvas site (short answer questions on directory nicknames), so that you can see if you are on the right track.

For the rest of the problems, you will create several files and then submit those to the course Canvas site.

NOTE: While I list the separate files you need to submit for each problem below, I am going to set up Canvas to *also* accept `.zip` files.

That is,

- you can submit each file to Canvas,
- OR, if you prefer, you may compress your files to be submitted into a single `.zip` file and submit that `.zip` file to Canvas.

Problem 1 - 10 points

Problem 1 is correctly answering the "HW 2 - Problem 1 - Short-answer questions on directory nicknames" on the course Canvas site.

Problem 2

Consider a directory containing the following files:

```
lab1.txt          lab2.txt          lab3.txt          OLDlab1.txt
OLDlab4.txt       ancestors.pl      save-it           slab-msrmts.txt
rules.rkt         lab45.rkt         lab754.pl         lab9.cpp
```

In a file named `hw2-prob2.txt`, put your name, and then your answers to each of the following.

2 part a

Write an `ls` command, that uses only a SINGLE argument including file expansion wildcard character(s) after its option, that would give a long-listing (including permissions) of only those files with a suffix of `.txt`

2 part b

It turns out that `save-it` is a directory. Write a command, that uses only a SINGLE argument including file expansion wildcard character(s) before the directory name `save-it`, that would copy ONLY those files with a suffix of `.txt` whose names before this suffix consist only of `lab` followed by 0 or more characters.

2 part c

Give the names of the files that would be moved into the parent directory by the following command:

```
mv *lab* ..
```

Submit your resulting `hw2-prob2.txt` .

Problem 3

Consider the directory `/Users/cbrown` . It contains a subdirectory `football` that contains:

```
game1-score.txt   f22-schedule.txt   past-seasons
```

..where `past-seasons` is itself a directory that contains:

```
f21-schedule.txt f21-scores      f20-schedule.txt f20-scores
```

In a file named `hw2-prob3.txt`, put your name, and then your answers to each of the following.

3 part a

Give the **absolute** pathname for the `f21-schedule.txt` file described above.

3 part b

Assume that you are within the directory `/Users/cbrown` (and that you have appropriate permissions). Write a *single* command, using a **relative** pathname, that would give the long listing (including permissions) for the file `f20-schedule.txt` described above.

3 part c

Assume that you are within the directory `/Users/cbrown` (and that you have appropriate permissions). Write a *single* command that would move file `f22-schedule.txt` described above into the directory `past-seasons` described above.

3 part d

Assume that you are within the directory `past-seasons` described above. Write a *single* command -- *without* changing your current working directory, and *without* using an absolute pathname -- that will give the long listing (including permissions) for the directory `football` described above. (Note that we want the permissions for the directory itself, *not* for its contents.)

Submit your resulting `hw2-prob3.txt` .

Problem 4

We mentioned two ways to reference the value of a shell variable during the Week 3 Lab -- you can put a `$` before it, or you can put `${}` around it. (That is, you can write `$`*my_var* or `${`*my_var*`}`)

Hint: if you want to put text "around" variable's value, the `${`*my_var*`}` notation is a good choice!

For example:

```
> pig=oink

> echo looky$pig

lookyoink

> echo ${pig}looky
```

```
oinklooky
```

In a file named `hw2-prob4.txt`, put your name, and then your answers to each of the following.

4 part a

Create a local variable `prob4` with a value of your choice.

4 part b

Write an `echo` command to display `$prob4 has the value` followed by the value of `$prob4` (note: I literally want to see a dollar sign followed by `prob4` at the beginning of this `echo` command's output)

4 part c

Write an `echo` command to display a string of 3 letter `o`'s immediately followed by the value of `$prob4`

4 part d

Write an `echo` command to display the value of `$prob4` immediately followed by a string of 3 letter `o`'s

4 part e

Write an `echo` command to display a string of 3 letter `o`'s immediately followed by the value of `$prob4` immediately followed by a string of 3 letter `o`'s

Submit your resulting `hw2-prob4.txt` .

Problem 5

Write a `bash` shell script named `backup-all` or `backup-all.sh` that meets the following requirements:

- Start this script with the line that is considered good style (and is a CS 279 course requirement), that specifies that this script should be executed using the `bash` shell
- After a blank line, put in one or more **comments** including at least the name of this shell script, your name, and its last modified date

After another blank line, write commands that do the following:

- it should create a new directory named `BACKUP` in the current working directory
  - (it is OK that you'll get a complaint if this directory happens to already exist -- we haven't yet covered the shell programming features needed to prevent that)
- it should set `BACKUP`'s permissions so that the owner/user has read, write, and execute permissions on it, but the group and the world/other have no permissions
- it should copy all of the non-directory files in the current working directory into the directory `BACKUP`
  - (and it is OK if it prints a message complaining about being unable to copy `BACKUP` itself over -- likewise, it is OK if it complains about being unable to copy over any other directory files that happen to be in the current working directory)
- it should echo to the screen a descriptive message indicating that it is about to show the current contents of `BACKUP`,
- ...and then it should output to the screen the current contents of `BACKUP`.

Also perform at least the following test of `backup-all`/`backup-all.sh`:

- do this test within a directory containing at least 3 non-directory files
- list the current contents of this directory, redirecting the results into a file `backup-all-test.txt`
- then run `backup.sh` in this directory, appending the results to the file `backup-all-test.txt`

Submit your resulting `backup-all` or `backup-all.sh` along with your `backup-all-test.txt` .

Problem 6

Run the `history` command -- see how it shows a listing of commands that you have done.

It turns out that following `history` with an integer <number> results in your seeing the last <number> of commands that you have done -- that is,

```
history 3
```

...shows just your last 3 commands.

You'll use this to create part of your output for this problem.

- Make a directory `279prob6`, and set its permissions so that you (the owner/user) has read, write, and execute permissions on it, but the group and the world/other have no permissions
- Demonstrate `279prob6`'s permissions with the following commands:

  ```
  echo "279prob6's permissions: " > 279prob6/prob6-perms.txt
  ls -ld 279prob6 >> 279prob6/prob6-perms.txt
  ```

  – (reminder: you use the `-d` option of `ls` when you want to see the name of the directory, not a listing of its contents. Using `-ld` lets us get the directory's permissions, not the permissions of each file within that directory.)
- Change the current working directory to `279prob6` (make it your current working directory).
- Create a file within `279prob6` named `prob6play.txt` that contains any contents you would like.
- Set `prob6play.txt`'s permissions so that you (the owner/user), the group, and the world/others have read and write permissions only.
- Demonstrate `prob6play.txt`'s permissions within the following command:

  ```
  echo "prob6play.txt's initial permissions: " >> prob6-perms.txt
  ls -l prob6play.txt >> prob6-perms.txt
  ```

- Now change `prob6play.txt`'s permissions so that you (the owner/user), the group, and the world/others each have a different set of permissions of your choice (any set is fine, as long as each level has a *different* set)
- Demonstrate `prob6play.txt`'s modified permissions within the following command:

  ```
  echo "prob6play.txt's modified permissions: " >> prob6-perms.txt
  ls -l prob6play.txt >> prob6-perms.txt
  ```

- You've done at least 12 UNIX commands at this point (and possibly more). Use the `history` command to figure out how many commands it has been since the command creating the directory `279prob6` (remember to count your `history` commands(s) as you determine this)!

  Then call `history` with an appropriate number argument (probably the number of commands since the one creating `279prob6` plus one), redirecting the output into `prob6-commands.txt`, such that all of your commands done for this problem, since the command creating the directory `279prob6`, are saved into `prob6-commands.txt`.

- (Note: don't worry if you have "extra" commands here -- I had them during my "test" run trying out this problem! `8-)`)

Submit your resulting `prob6-perms.txt` and `prob6-commands.txt` .