CS 279 - Homework 9

Deadline:

Due by 11:59 pm on **SUNDAY**, November 2.

How to submit:

Submit your files using `~ah270/279submit` on nrs-labs, with a homework number of `9`, by the deadline shown above.

Purpose

To practice with commands such as `ls`, `head`, `tail`, `diff`, `gzip`, `gunzip`, `sed`, and `tar`, along with setting one's command prompt, using the `BASH_REMATCH` array, and practicing some earlier concepts as well.

Important notes:

- **Each** bash shell script that you write is expected to include a descriptive opening comment block including your name and the last modified date.
- It is possible that your answers may be collected and posted to the course Moodle site.

The Problems:

Problem 1

In a file `hw9-1.txt`, include:

- your name
- the part you are giving an answer for
- an appropriate SINGLE command for each of the following (note that SOME of these involve pipes):

1 part a

Do all three of the following:

- list all files, including "hidden" ones that begin with a dot (".")
- identify directories with a /, executable files with a *, and symbolic links with a @ (the last one seems to work on MACs, but not on Fedora, but that's OK)
- List them in chronological order, latest first

1 part b

Display only the first five lines of the file `.bash_profile`.

1 part c

Find all the differences between the contents of the files `tweedledee` and `tweedledum`, then do a search of those lines for the character string `"twins"`.

1 part d

Read the contents of the file `SSU.txt`, change all the appearances of the text string `Sonoma` to the string `Humboldt`, and save the resulting text to a new file named `HSU.txt`.

1 part e

Interestingly, it turns out that you can use the `-c` option with the `head` command as well as with the `tail` command -- and, analogously, with the `head` command it indeed causes the first specified number of bytes to be displayed.

Use this to write a command that will display *just* the permissions of file `lookity.txt`. (For this problem, it is OK that its output does not include a newline at the end.)

1 part f

How can you get just the number of lines in a named file? There are numerous ways, but here is one approach. `wc` includes the file name if called with a file name -- but it does not if called with standard input. Write a command with a pipe that pipes the contents of `lookity.txt` to `wc` such that only the number of lines in `lookity.txt` is the result (albeit preceded by blank space).

OPTIONAL: also include your alternate favorite way to just get the number of lines in `lookity.txt`.

1 part g

You want to archive the directory `279tools` and all of its contents. Write a command that will do so, resulting in the archive file `279tools.tar`

1 part h

You now want to compress `279tools.tar`. Write a command that will do so, using `gzip`, that will also let you know how much this file was compressed.

1 part i

Your result from part h has been copied or e-mailed to another directory somewhere. Assume the current working directory is this other directory with this copy of part h's result. Give the command to now uncompress it.

1 part j

Now give the command to restore/expand the directory `279tools` from the result of your command in part i.

1 part k

Now give a command you could use to recursively list all of the contents of this restored/expanded copy of `279tools`. (I can think of at least two different commands for doing this, and there are probably more! Choose your favorite or one that you want to practice with.)

1 part l

Write a command that would set one's command prompt so that it becomes a three-line command prompt including the current date on the first line, the current time on the second line, and then `"> "` on the third line. You can choose the format you prefer for the date and the time, but here's an example resulting prompt that meets the criteria:

```
Fri Oct 31
10:40 PM
>
```

1 part m

Write a `sed` command that will add four blanks to the beginning of each line in file `lookity.txt`, putting the result in `shift-look.txt`. (Hint: think of it as substituting 4 blanks for the beginning of each line.)

1 part n

Give the last 3 lines of every file whose name ends in `.txt`

Submit your resulting `hw9-1.txt`.

Problem 2

Write a bash shell script `summary.sh` that meets the following specifications. It should:

- if there are no command-line arguments, ask for a file to be summarized; otherwise, it assumes that the first command-line argument is a file to be summarized. (It can simply ignore any additional command-line arguments.)
- check to ensure that the file to be summarized exists, is a regular file, and is readable; if any of these are not the case, then output an error message and exit with a non-zero exit status
- output the following for this file, formatted in a readable manner of your choice:
  - its permissions (given as shown in its long listing, but NOT the entire long listing for this file, however)

- the number of lines in this file
- IF there are more than 6 lines in the file, the first 3 lines of this file, a `...`, and then the last 3 lines of the file
- OTHERWISE, just the first 10 characters, a "...", and then the last 10 characters. (If there are too few characters, it is OK if there is some or complete repetition here)

Submit your resulting `summary.sh`.

Problem 3

3 part a

In a file `hw9-3.txt`, put your name, and:

- a regular expression, with a subexpression, that would match an HSU CS course prefix and number of the form `CS XXX` where `XXX` is any 3 digits, and the `XXX` is matched by a subexpression.
- a `for`-loop that will use a backquoted command to help it loop through all of the files whose names end with the suffix `.txt` in the current directory, outputting `"found: "` followed by the file name for each such file
- a code fragment setting a counter variable to 0, followed by a `while` loop that reads each line of a file `$file`, incrementing the counter variable for each line and using it to output `"touched "` followed by the line number and a blank followed by the line for each line. (Hint: remember to use `let` when incrementing your line counter variable.)

3 part b

Now write a little bash shell script `get-cs-num.sh` that looks to see if its first command-line input contains an HSU CS course prefix of the form describe in part a. If it does, it uses the `BASH_REMATCH` array to simply display just the course's number -- for example:

```
> get-cs-num.sh "This is CS 279 Homework 9"
279
```

If it doesn't, it outputs nothing.

Odd things to note:

- you know how we had to escape the parentheses in a regular expression's subexpressions used with `grep`? We **don't** need to escape those parentheses in a regular expression used with =~ within a bash shell script!
- HOWEVER, because we know that quotes of any kind seem to be problematic with =~'s regular expression, note that you CAN escape a blank in this setting by preceding it with a backslash -- e.g., to match `"moo oink"` in `$stuff` I could use:

```
if [[ "$stuff" =~ moo\ oink ]]
```

3 part c

Now write a bash shell script `log-refs.sh` that meets the following specifications:

- for each `.txt` file in the current directory, it reads through each line of such a file,
  - and for each line of a `.txt` file containing at least one reference to a CS course `CS XXX` -- as described in part a -- then it appends that file name, `": line "`, and the line number of that reference in a file `csXXX-refs`, where `XXX` is the course number of the first CS course reference found in that line
- for example, if line 3 of file `looky.txt` was:

  ```
  The rain in CS 279 stays mainly in CS 100
  ```

  ...then the line:

  ```
  looky.txt: line 3
  ```

  ...would be appended to the file `cs279-refs` (but not to `cs100-refs`).

Submit your resulting files `hw9-3.txt`, `get-cs-num.sh`, and `log-refs.sh`.