Before you begin, make sure you have access to a Linux-like command line. See the website for help getting set up.

1. Using a single `echo` command, print the following to the terminal:
   ```
   line1
   line2
   no newline here:\n
   line4
   ```
2. What variable contains the default shell? What variable contains the currently running shell?
3. List all the shells available on your system.
4. (a) Create a variable called `v1` to hold the string `snow` and a variable called `var2` to hold the string `man`
   (b) Using a single echo command and only these 2 variables, print `snowman`
   (c) Using a single echo command, print `manatee`, with the `man` part coming from your variable
5. Write a command asking a user to enter his favorite movie and store it in a variable.
6. Write a command asking a user to enter his credit card number and store it in a variable. Ensure the numbers are not printed to the terminal as the user types.
7. Write a command to print all files in the current directory whose name starts with `a` and is 2 characters long.
8. Write a command to remove all object files in the `src` directory (hint: an object file ends in `.o`).
9. How would you match a file whose name is literally `*.txt`, without matching other text files in the same directory?
10. Create an alias called `greetme` that prints `Hello,` followed by your name when run.
11. Create an alias called `greet` that prints `Hello,` followed by the value of the `$name` variable when run (this is tricky: think about the type of quotes you use and when variables are evaluated. Make sure to test by changing the value of `$name` after creating the alias).
12. Create an alias that removes all object files in the current directory, clears the screen, and then prints `"Cleaned"`
13. How do you view the exit code of the last run command?
14. Write your current working directory to a file called `files.txt` without using a text editor.
15. Append all the files in the current directory, sorted by file size, to `files.txt`.
16. `sort` reads lines on `stdin` and prints them sorted alphabetically to `stdout`. Print the contents of `files.txt`, sorted alphabetically, to `stdout` by redirecting the `stdin` of `sort` from the file.
17. Download the list of animals from the website with `$ wget www.cs.umd.edu/command_line/animals.txt`
18. `cat` the list of animals and pipe the output to the input of `fgrep` to only print the turtles
19. Pipe the output of that command to the `sort` command to sort your turtles by name
20. View `animals.txt` using a pager

21. `fgrep` has an exit code of 0 if the target string is found (like `turtle`), or a non-zero exit code if it is not (like `camel`). Write a command than prints out all of the `hyena`s in `animals.txt`, or `None found` if there are none.

22. How do you repeat the last command ran? The last `cat` command ran?

23. What does `rm !*` do? `rm !ls:3`?

24. Create a variable called `name` holding your full name. This should have at least 1 space in it.

25. Create a file with the `touch` command whose name is your full name, using the variable from the last question. Be careful with the spaces?

26. Use brace expansion to list all the files in the directories `classes/cmsc216/notes`, `classes/math241/notes`, and `classes/comm107/notes`.

27. What does `mv {,old_}file.txt` do?

28. Use command substitution to save your current working directory into a variable called `curdir`.

29. Assume `dirs.txt` is a list of directory names. What does `mkdir $(cat dirs.txt)` do?