clmystery (Command Line Mystery)

1. Download the clmystery zip and unzip it to the current directory. To begin the challenge, use the `cat` command on the "instructions" file. 'cat' stands for "concatenate", although many use it to dump the contents of a file in unix.

   As the instructions say, *usage of text editors are forebidden*. We'll also forbid the use of the commands `less` or `more`.

2. Now that you have background, it's time to use terminal skills to solve the mystery! Enter the mystery directory and look around. The data is somewhat unstructured so to get a feel for what each file looks like, instead of `cat`'ing each file, and dumping tons of content to the terminal, try using `head` and `tail`. `head` and `tail` dump only the beginning or end of a file, respectively.

3. As you might have noticed, the `crimescene` file contains snippets of *Alice in Wonderland*. Unfortunately, the cops got it all jumbled together somehow. Given the clue in the instructions, we need to somehow search for string CLUE in the file, without a text editor and avoid the rest of the *Alice in Wonderland* junk. The command of use here is grep. Try

   ```
   grep "CLUE" crimescene
   ```

   and you should get a listing of matching lines. *Did you find clues? There should be three.*

4. One clue should have a name, a witness! The police put the people they interviewed in the `people`. Try using `grep` to find the witness. *what command did you use?* Also, you can filter based on part of the clue, based on the gender of the witness. Use this to decide who to look up next. *You should have two possible witnesses.*

5. The police visited the people in the `people` file at their residences and put down information in their logs, again, full of junk. For the *two* matches you obtained from 4, go to the `streets/` directory and look for the filename that matches the street. We want to somehow pick out the file line next to the street name, again, without the editor. Here is where piping will come in handy. You might have looked up the manual for `head` or `tail`. Head has an option `-n` that allows you to print a subset of the file. For example,

   ```
   head -n 100 file
   ```

   will output the first 100 lines of `file`. This is useful, but it leaves a lot of junk in the terminal. In the meantime, `tail -n 1` would print out *just* the last line of a file. Could we combine these to make the terminal just print the one line we want? Try this:

   ```
   head -n 100 file | tail -n 1
   ```

If you run this command with `file` replaced by the street file you want to search, and 100 by the file line you want, you should see only the line you want. This is called piping. Piping, using the | pipe, essentially takes the output of one program, here `head`, and outputs it to the command on the right of it, here, `tail`. `head` reads the first 100 lines of `file`, then outputs that to `tail`. `tail`, taking in 100 lines from head, outputs just the last line of the 100, thus, printint only the 100th line, and nothing else. Another example of piping is

```
cat file | head -n 100 | tail -n 1
```

Here, we have three commands, two pipes. First, `cat` simply outputs the entire `file` to the stream, `head` takes the first 100 lines and outputs only that, then tail takes those 100 lines and outputs just the last. This shows how the output changes accross the "stream" from left to right. Piping is a powerful tool that allows one to combine many simple programs that work with text files to perform powerful search and filtering operations. You'll use this in the next few steps.

6. The last step should have given you an interview number. Find the corresponding interview in the `interviews/` directory and output it using `cat`. You should find another clue if you chose the right person. If not, try finding the other person's street, and their interview using the same steps in 5. You should have another clue, specifically a description of the getaway car! Look at the beginning of the `vehicles` file using `head`. Use the command `wc` with the option `-l` to count how many vehicles there are in this file.

```
cat vehicles | wc -l
```

here, we are using `cat` to output the file, and the pipe to pipe the text of the file to `wc -l` which counts the lines. ***Given hints about the vehicles see if you can find 35 or less matching vehicles to ones that matched the description by the witness. Use wc to count them.*** Realize that you can use an arbitrary number of `grep`'s separated by pipes. For example,

```
cat file | grep food
```

will output lines in file that match food, but

```
cat file | grep food | grep candy
```

will match lines that match food ***and*** candy, that is, have food and candy both on the same line. ***Which command did you use?***

7. You can filter the list down further by nothing the clue from the witness said the License plate began with a string and ended with a number. Here we can use "regular expressions." It turns out that grep actually doesn't just search for strings in a file, it searches for lines that match a regular expression. For example say you wanted to find lines that have either the pattern "aaab" "ab" or "b", that is, strings that contain an arbitrary number of a's and one b at the end. One could use the regular expression

```
a*b
```

The "a" character matches an a, the asterisk (*) with that a in front means that "match a string that contains any number of a's, including none", and the "b" matches a "b". Thus,

```
cat file | grep 'a*b'
```

will find any file that contains "b", "ab", "aab", "aaab", and so on. *Be sure to put quotes around a regular expression on the command line!* What if you want something more general, any string that ends with b? This is where the period (.) in regular expressions come to use. (.) matches *any* character. So

```
cat file | grep '.*b'
```

will match lines with the strings "ab" "cb" "gdjsb" and so on. *Now, modify one of your grep's on the vehicles file to filter more specifically for the license plate description from the witness. HINT: You might not need *! You should have 15 or less vehicles if you do it right.*

8. EXTRA: Solve the mystery! Recall the clues from the `crimescene`. Using the one of the clues you haven't used, you should be able to narrow it down to four suspects from the 15 you have obtained. You can do this either by eye/hand or with an extra pipe and grep. Finally, checking the gender of each of them (at least for me, it wasn't obvious, run against the `people` file if you aren't sure), you can use the last, unused clue from the `crimescene` to figure out which one of the remaining two had dunnit. HINT: Use `cat`. `cat` originally was for concatenating files. For example

```
cat haystack1 haystack2
```

will output the contents of haystack1 to the terminal and then output the content of haystack2. So,

```
cat haystack1 haystack2 | grep needle
```

will output lines matching "needle" in haystack1 and haystack2. Use this to figure out whodunnit in two commands. Finally, if you think you know who did it, `cat` the "solution" file in the clmystery directory for instructions on how to check it.