The Command Line Murders

(2 points.) To start, run `python new-homework.py [language] cli-murders`

Background. The Unix shell provides a convenient set of tools for searching, sorting, filtering, editing, and generally munging any kind of text—program output, CSV files, LaTeX source code, or anything else you might need. The Unix philosophy is to provide small, reliable programs which do one task well, then provide tools to connect and combine these tools to do almost anything. You can easily write your own scripts to accept output from other programs, or process the output of your programs with other shell commands.

A famous example is Doug McIlroy's shell command to read a text file and output a list of the n most frequently used words in the file, along with their frequencies. He wrote the script in response to an article by Donald Knuth, which espoused literate programming by solving the problem in several pages of detailed, well-documented Pascal. McIlroy's reply uses just six commands:

```
tr -cs A-Za-z '\n' | tr A-Z a-z | sort | uniq -c | sort -rn | sed ${1}q
```

By using *pipes* to send the output of one command to the input of the next, these small programs can solve the problem with far less work than Knuth's elegant solution. Here's McIlroy's explanation:

> If you are not a UNIX adept, you may need a little explanation, but not much, to understand this pipeline of processes. The plan is easy:
>
> 1. Make one-word lines by transliterating the complement (`-c`) of the alphabet into newlines (note the quoted newline), and squeezing out (`-s`) multiple newlines.
> 2. Transliterate upper case to lower case.
> 3. Sort to bring identical words together.
> 4. Replace each run of duplicate words with a single representative and include a count (`-c`).
> 5. Sort in reverse (`-r`) numeric (`-n`) order.
> 6. Pass through a stream editor; quit (`q`) after printing the number of lines designated by the script's first parameter (`${1}`).

Task. To learn the basic shell searching and manipulation commands, you will complete the Command Line Murder Mystery, available at `https://github.com/veltman/clmystery`. Use Git to clone the repository. Do **not** clone the repository inside your assignments repository—having a Git repository inside another Git repository is a recipe for confusion. Place it elsewhere:

```
cd ~/s750
git clone https://github.com/veltman/clmystery.git
```

In your assignments repository in the `cli-murders` directory, create a text file `STEPS.txt` detailing the steps you took and commands you used to reach the solution, explaining the purpose of each command along the way. This is what you will submit as your solution to the problem.

(Mac and Linux users can use their OS's built-in shell; Windows users should use the Bash shell that comes with Git, install Cygwin, or learn to use PowerShell, which has a completely different syntax and style.)

Pro tip: The ExplainShell website lets you enter a command and have its meaning explained to you, piece by piece: `https://explainshell.com/`

Requirements.

☐ Complete the Command Line Murder Mystery. Follow the instructions in the provided `instructions` file. As it states, ***don't use a text editor*** to view any files but the instructions, cheat sheet, and hints—you ***must*** use the shell to find the content you want inside the other files.

☐ In `STEPS.txt`, explain the commands you used to reach your solution.

☐ You may look up references for each shell command you're interested in, but you may not cheat by looking up the solution or anyone else's methods of solving the mystery.