1. Briefly describe the design of the program. How many processes and how many pipes are created with your solution? How does data flow from process to process?

My program has 5 processes and 4 pipes. The parent process forks 4 child processes. The four child processes each execute one command: find, grep, sort, head. Lastly, the parent process spawns and manages the other 4 processes. The data flows between the processes by first calling a process(find) that inputs from a designated directory(bash-4.2) outputting to a pipe(1), this pipe(1) is then read by the following process(grep) that outputs to another pipe(2), this continues in a similar fashion with the sort and head processes using pipe(2) and pipe(3) respectively as inputs, and pipe(3) and pipe(4) as outputs. Lastly, the parent process takes the output generated by the final process which it inputs from pipe(4) or the output of the previous process. This is read to the terminal.

2. How did you test and debug your solution?

I programmed the processes sequentially and outputted the outputs to the terminal instead of the proper pipes for initial testing. When this outputted correct values I then outputted to the pipe. I then followed this same procedure for the next process where I inputted from the pipe and outputted to the terminal to check the outputs of that process. This was repeated for all the processes. If I got terminated by signal 13, I checked the piping of the previous process and deleted the process I was currently writing and tried again since I had clearly misused the pipes and it was easier to try again than to find and fix the issue.

3. When he was head of Bell Labs, Doug McIlroy summarized the "Unix philosophy" as follows:“Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.” How do pipes contribute to the Unix philosophy?

Pipes contribute to this by allowing connections between very specialized small programs. These programs do one thing very well and then the output is passed. Pipes are also very useful because the easily pass text streams into a process that then operates on the stream. The process can then use them to pass a text stream out as well. This continues for as long as it needs to and the only linking between the tiny specialized processes is the pipes themselves. The input and output of processes can easily be changed by changing the pipes they are using.