In writing the two phonebook scripts, one in bash and one in C-shell I recognized some similarities, but also many differences. At first glance, the scripts look like the bash script has more lines, however upon closer inspection, they are actually around the same number of lines. This is because of an error I encountered when writing the C-shell script.

When working with the bash script for case 6, I was able to separate my awk command into multiple lines for easier readability. So I ended up having linebreaks in my script for splitting the date, getting the month/year, and then comparing and printing the line if it matched. This was all surrounded by single quotes so they were recognized as one command despite the linebreaks. For the C-shell script, however, there would be an error whenever I had linebreaks in the awk command, even with single quotes. To fix this error, I changed the awk command to use one line and no linebreaks and the script was then able to recognize the single quotes. This error may have been due to the way that C-shell recognizes the quotes as they are different from bash.

In writing the bash and C-shell scripts, I found out that they are similar in the sense that I was able to structure them the same way. Both of them first check for the correct number of arguments (1 which is the filename) and then check that the file exists. The syntax for these was different. For example, the if statements in bash are noted that they are done with *fi*. For C-shell, the if statements are noted that they are done with *endif*. Another difference in this portion of the scripts was that bash uses the brackets to do comparisons or tasks like file opening, while C-shell uses parentheses to do comparisons or tasks.

For the next part of my scripts, I printed the menu using *echo*. This was done pretty much the same between both bash and C-shell where I just used echo to print out the menu. The differences between this part of my scripts were that to loop the menu (so that the user would constantly be prompted until they wanted to exit the script), the bash script's while loop was *while true; do*, and the C-shell's while loop was *while(1)*. Another difference was that to read the menu choice from the user, bash used the *read* command while C-shell used the *set choice = $<* to read the input.

To handle each menu option, I used a switch case for both shells. The syntax for the switch cases are different in bash and C-shell. For the bash script, the switch case was defined with *case $choice in* followed by the cases. C-shell on the other hand, uses *switch ($choice)* instead and uses *case* to define the different cases. In bash, the double semicolons are used as the break statement signifying the end of the case. For C-shell, instead of using double semicolons, *breaksw* is used to break and signify the end of a case.

For the first 4 cases, I had to sort the file according to the first/last name. To do this, I used the *sort* unix command and used *-t* to separate the columns by spaces. This way I could sort the first column (first names) by specifying *-k1* or the second column (last names) with *-k2*. Because the only two things I really cared about for the first four functionalities of the script were the first and last name (which were separated by a space), using *sort -t ' ' -k[column#] "$filename"* was able to be used for all of these cases. In the cases where I wanted to sort the names in reverse, I would also use the *-r* tag to sort in reverse. I was able to use this in both bash and C-shell.

For the 5th case there were two main differences between the scripts. The first main difference was that in the bash script, I read in the last name that was to be

searched for using *read -p "Enter last name: " last_name*. On the other hand for the C-shell script, I read in the last name using echo to print out the prompt and the using *set last_name = $<*. The second main difference was that the *awk* command in bash couple be broken down into multiple lines for easier readability. For C-shell, I ran into an error saying "Unmatched ' ' '" whenever I tried to have the command in multiple lines which made the readability a bit more difficult for the C-shell script.

The 6th case, was similar to case 5 for things like taking in input and using *awk*. Aside from the differences I mentioned earlier in case 5, case 6 also had two main differences. The first main difference was that to get the length of the user input, I could use # to get the length of the user input (this was used to identify whether the user inputted a month or a year. In C-shell, to get the input length, I used *expr* to get the length. The second main difference for case 6 was the way that the comparisons were done. In the bash script, the comparison had to be done in the square brackets and to check if the length was equal to a number, *-eq* was used. C-shell on the other hand was done in parentheses and instead of using *-eq*, double equals == was able to be used.

In case 7, (the exiting script case), they were done the exact same aside from the case definition where bash used *7)* and C-shell used *case 7:.* In both, I implemented this case by echoing "Exiting…" and then followed by the exit command.

The last difference I noticed between bash and C-shell scripts is that for any other case, in bash, it was noted using the *. So for the remaining cases, the code is **)* to where I would echo "Invalid Option. Please try again". For C-shell however, instead of using the *, *default:* was used.

After creating the phonebook script in both bash and C-shell, I would say that bash is better. While many of the functionalities were able to be done in the same number of lines with only slightly differing syntax, the main reason as to why I say bash is better than C-shell is due to the error I encountered with the awk command being unable to written in multiple lines. Because I was able to separate the awk command in the bash script, it was much easier to read and debug. Especially when the awk command had several parts, like splitting the date, comparing the month/year to the split part of the date, and printing the line.

Another reason why I prefer bash over C-shell is because the syntax feels a bit easier to remember and read. When I look over my bash script, it is very easy to understand what each line does, and to recognize which lines are actually performing tasks like sorting. Since the cases are defined by *#)* and the end of the case is marked with *;;* this makes it easy to identify lines like *sort -t ' ' -k1 "$filename"*. On the other hand, when I look at my C-shell script, it is filled with many more words. For example a case in my C-shell script is *case 1:* followed by the *sort* command and then *breaksw*. While this doesn't seem like too big of a difference, the jumble of letters makes it a bit harder to pinpoint immediately compared to the syntax of bash.

The last reason I prefer bash over C-shell is its more intuitive syntax for reading user inputs. For example, bash was able to read the user input in one line and with the command *read -p "Choose an option" choice*. C-shell on the other hand, required two lines using one to print out the prompt, and one to read it. When I try to use read in the C-shell, I get the error "undefined variable". The excess lines required for C-shell seems like a minor inconvenience (and it is), but I think that being able to read the user input in a single line makes writing the script more efficient and easier to create.