Conclusion:

When testing our gifting algorithm, we discovered that simply adding tickets to interactive processes or removing tickets would influence longer running processes. If a new process was added, it would start with only 500 tickets, and even if it would be considered interactive, it would take time for it to accumulate enough tickets to be able to run consistently, while longer-running interactive processes would already have more tickets.

By using the nice value and interactivity value to set the tickets in each process, rather than add or subtract tickets, newer interactive processes had a more equal chance at being selected similarly to old interactive processes. This made interactive processes seem responsive more consistently. By having processes all have different amount of tickets, we can see which processes get more attention than others. Meaning, the amount of CPU each processes will use will be proportional to the process' tickets.

When debugging to ensure that tickets were being set correctly (simply using a printf() to print the number of tickets every time a user process is scheduled), we observed that when the FreeBSD shell would wait for user input, it would have a high amount of tickets. This is expected, since the shell program is blocking on I/O, meaning that it is an interactive program. On the other hand, when the enter key was held down, the shell's ticket count would decrease over time. This also makes sense, since now the shell is not blocking on I/O since it is continuously receiving input. The program would now be considered CPU bound, and thus less interactive.