ECE326 Lab 1

Q1. If the system cannot print up to 100!, what will be the reason?
If the system cannot print up to 100!, the reason will be that the system ran out of memory.

Q2. Find out the maximum number possible on your system. Provide reasons.
When using recursion, the maximum number possible on my system was 997. This is because recursion uses a stack which uses a lot of memory for large numbers. When using iteration, the maximum number possible on my system is infinite (or it is very large). My iteration program ran for 5 minutes calculating 200,000!+, and it was still running when I forced it to end. The iteration algorithm has a larger maximum number possible on my system because it requires a lot less memory than recursion. I.e., only one value needs to be stored and replaced–this requires a lot less memory than maintaining a stack.

Q3. Check all boundary conditions (Hint: no factorial for -ve number).
I added a check for boundary conditions to the factorial functions.

Q4. Calculate the total time (ideally CPU time, but wall clock is fine) required and report it.
I used the Python time library. The CPU time for recursion was 2.9802322387695312e^-05 seconds and for iteration was 7.152557373046875e^-06 seconds.

Q5. Try to optimize the computation (Hint: 9! Is simply 9*8!) and calculate the total time again. Did the time reduce? Explain both Yes and No cases.
Iteration is the optimized computation over recursion. The times for recursion versus iteration are specified above. Yes, the time is reduced because there are less ‘overhead calls’ needed in iteration than recursion. I.e., iteration only requires updating one variable value whereas recursion requires maintaining a stack (push and pop). The No case occurs when the time did not reduce (i.e., it stayed the same) or it was slower. In the first situation when the time stays the same, this would occur because O(n) is the best time for factorial algorithms. My iteration algorithm was already O(n) time so any optimized solution will also be at best O(n) time. The second situation where the time is slower should not occur for an optimized computation.