I spoke with Bryan to get clarification on how things should be implemented. I have two main tasks which are to change the implementation of General.py so inputs are activators and to make sure the output does not hide a gate.

A potential problem is that when the inputs are only b and c, IN1 will be b and IN2 will be c rather than IN2 as b and IN3 as c.

I think I was able to get the program operational. However, it is strange that I did not need to change the inputs to activators to make it work. I tested it on some test circuits and it to function as expected. One thing I noticed is that the more gates you go through, the more muddled your signal is, which is expected and causes unexpected behavior of the overall circuit. When going through gates, the muddling might make the protein concentrations decrease. Then when you hit a not, since it is so low, it will shoot up. Then if it is NORed with something of an intermediate value, the high one will overpower it. It seems the extreme high and low always tend to overpower intermediate values.

A possible idea would be to, when making circuits, make the subcircuits first, then determine which ones have the most distinct highs and lows, then choose that one to integrate into the overall circuit. Actually, you may run into a global vs local optimum problem here because you might use your best repressor early on because it is useful there, and not be able to use it later on where it may have been even more useful.

Notes: See if there is a way to extract the concentration of each item at each stabilization point and graph it in a more understandable way. A way that would show when it is expected to be high or low and what it actually is. I must ask if I can get the files with the inputs, repressors, and outputs so I can work on integrating them into the function.

I will need to generate the libraries using the value that I have varying it from 1/5 to 5x current values. I will probably just use a random number generator to do this.

I still need to comment the code well.

The sorting of the gates for the DAG string representation needs work. I don’t know how to properly sort the gates in order. Maybe I can do some sort of depth first search starting from the output then reverse the order.

I generated 9 random libraries and made graphs from each of them for ((a.b).0) and library 7 seemed most promising. I used the items in each library just in order from top to bottom until I don’t need any more genes. Libraries 4 and 7, although not great, seemed like the best choice for “(a.b)”.

I am going to work on a way to sort the gates in a DAG so the DAG’s netlist will print in order.  
I completed a way of sorting the gates.

Tomorrow I will work on commenting the code to improve its readability. I will also look into making more realistic values for the gates and inform Bryan on my progress.

Other things on the TODO list include making graphs that can make it easy to visualize the level of a protein and whether or not it is supposed to be high at that point and making this program go from NETLIST to gates rather than string implementation to gates.