Daniel Tweed, 6791717

Part 1 –

Parts 2 and 3 have the same outcome as part one, but with different approaches. Since Dr. McLeod said to user higher level approaches, I did not implement this lab as low level as instantiating the actual flip-flops, but not as efficiently and high-level as the approach looked for in the later parts. This made the HDL more complicated, and probably less efficient, even after optimized by Quartus, but it was effective and did the job. No real issues encountered.

Part 2 –

Implemented using the skeleton provided in the lab write up, and no significant issues, except Quartus complained that **G** was an illegal name.

Part 3 –

No issues encountered in implementing this. Did not use an actual shift register module, just shifted a **reg** in the main module. This could have been done using one register and just kept track of if we were counting 1's or zeroes, and reset if it switches.

Part 4 –

I found that I spent a lot of time drawing the state diagram for this question, and, although useful practice, was an overly large amount of time to commit. Instead, approached as I would have if coding in a procedural language, and made the necessary modifications. This made the Verilog likely more complicated, and had to have one large **always** block, as had multiple events driving the same registers. This worked, but I can see this is likely not the best (most efficient?) way to implement a solution to this problem.