Summary:

The way my program executes follows as such: Create and start all Customer threads, Bank Teller threads, and Loan Officer Threads; Bank Tellers and Loan Officers prompt Customers that they are ready; Customers decide what task they want to do; If a Teller is available, a Customer confirms the Teller's notice, prompting them with a ready notice, to which the Teller confirms as well, thus creating synchronization and allowing Customer and Teller to perform the task the Customer chose; Similar steps occur with the Loan Officer; If no Tellers or Officers are available, Customers wait their turn before a Teller or Officer declares availability; Rinse and repeat; Once all Customers have performed 3 tasks each, they record the final results, and are rejoined in main; Once all Customers are rejoined, all other threads are halted; Before program exit, recorded final results are displayed in a simulation summary.

From my experience with this project, I cannot really say I had much difficulty in getting the design and code created. I approached the design phase with a very careful mindset, to where every interaction between customer and teller/officer was met with a "action performed, did you receive?" kind of double checking to ensure all thread interactions were synchronized and not crossing over one another. However, I found it harder to progress with designing any semaphores or functions without actually testing it out with code first. So, I ended up creating a rough idea of how I wanted the structure to look, and filled in all the gaps as I progressed through the coding phase, doing old fashion trial and error. The most tedious part was designing the semaphore interactions between customer/teller/officer in a manner that was structurally similar, and performed in roughly the same time as an additional precaution. To my luck everything functioned well and in order on my first dry run, so having the most logic-intensive part out of the way was a relief, and made wrapping up the rest of the program rather smooth. Granted in the end, all of my "just to be safe" decisions while coding might have produced some unnecessary semaphores or actions, but I was able to consecutively provide dozens of simulations that did not have any synchronization or logical issues, so I wasn't about to start messing with code that wasn't broken.

In the end, semaphores are a really simple concept to grasp. Mimicking interactions made at a bank was straightforward to recreate, as the majority of semaphores were just back and forth "I did this, do you acknowledge?". The toughest part was figuring out the right order to put them in so data was passed and manipulated properly with all the threads running at once. Once I got the customer's semaphore blocks created, I pretty much had the teller's and officer's created too, as all I needed to do was invert the acquire/release signals and change out commands with the ones related to the appropriate role. Once the semaphore blocks were finalized, I tweaked and cleaned up the rest of the program to the best of my ability. And after a couple dozen more runs and checks with the output ordering, everything functions flawlessly.