Class: Fall 2022 - CS 4348.502
Student Name: Ahmet Mert Buyulu
Net ID: AMB190003
Date: 11/03/22

Project 2 - Write-up

1) **Question**: How did you approach the project?

   Answer:
   Since the purpose of this project to demonstrate the understanding of how multiple independent threads can interact with each other via Semaphores, I first made research on what a Semaphore is, how it works and creates synchronization between multiple threads, and the language and the libraries I wanted to use to implement the project during the first week of its assignment to the class.

   Having said that, I couldn't work on the project for the next two weeks due to exams and other workloads coming from my other classes. Thus, I started coding the project during the week it was due. However, understanding how the interaction between a customer and teller should be and semaphores should be used to keep the interaction in the order I should be early on the project, I was able to finish the project within the next 12 hours after its original due date. [Note: I'm using my one-time free late submission pass on this project assignment.]

2) **Question:** How did you organize the project? Why?

   Answer:
   I organized the project by determining which action I should take in each step through the process of implementing the project to complete it with as minimum confusion and stress as possible.

   I started off understanding what resources needed to share between customer and teller threads, identifying the number of semaphores the project needs, and the issues regarding the synchronization between customer and teller threads as well as the resource limitations (e.g. two teller threads, at most, can only access the safe on any given time and each of the tellers cannot serve more than one customer concurrently).

   Once I really understood the requirements, I started implementing the project using three tellers and customer threads to keep the program simple. After completing all the project requirements, I increased the number of customers going into bank to be set to 50 as described in the project instructions. To add on, I checked the output of the program to see if the synchronization between customer and teller threads works as intended. Finally, I concluded that the program is ready for submission.

3) **Question:** What problems did you encounter?

Answer:
1 – The program was not terminating successfully after no customers left to serve, and the bank is close, requiring me to force quitting the current execution via keyboard interrupt (ctrl + c).

4) **Question:** How did you fix them?

   Answer:

   1 – I researched about why the program may have not been terminating successfully once the main thread finishes its execution. I realized that sometimes main thread may resume executing before all its sub threads finish executing. Although, I had semaphores in place to control the synchronization of all the threads, I needed to use the join method in the threading library on each of the teller threads. This enforced main thread to wait until all the teller threads finish executing, resulting in successful termination of the program.

5) **Question:** What did you learn doing this project?

   Answer:

   1 - I learned what threading library is used for and how to work with it.
   2 – I learned how semaphores work on basic level, their types (binary, counting), and how they can be used to restrict access to shared resources (safe, door, teller registers, etc.).
   3 – I learned how to work with threads and how variables can be passed between individual threads.
   4 – I understood why joining multiple treads is useful for preventing possible deadlock situations where the program doesn't successfully terminate after the bank is closed.
   5 – I learned the advantages/disadvantages of using semaphores vs lock and what scenarios are more suitable for each of these concepts.

   Question: If you did not complete some feature of the project, why not?

   Answer: I did complete all the requirement the project asked for.