Berk Dağlı 150115039

Hüseyincan Erbayraktar 150114015

Ali İbrahim Mercan 150115038

CSE4034 PROJECT 1 REPORT

In our project, every customer thread takes its own “C_Account” structure as parameter which indicates customers id, number of allowed operations, number of allowed reservations and a mutex which is only used when waiting for a condition variable to be signalled from main thread between simulation days. Customer threads prepare tasks using prepare_transaction() function for the sellers until it is end of the day or there is no allowed operations left on that day. What function prepare_transaction() does is, that it generates a random number from 0 to 2 to represent the operation and if that operation is not valid, flag is set to 1 and another operation is selected until it is valid. After a transaction is created, customers loop through the sellers until they can lock. The connection between the customer and the seller is provided via “Seller_Request” structure. That contains customer id, seller id, transaction prepared by the customer, mutex to lock the seller and a condition variable to sleep the customer thread until seller thread finishes its job and sends a signal to wake it up. When the seller threads are created, they are given their own Seller_Request structure as parameter. When the customer thread copies its transaction to corresponding sellers Seller_Request structure, seller thread handles the transaction and makes changes to the products which are represented with “Product” structure. When the changes are to be made, seller locks the corresponding Product mutex. If the transaction is valid, seller thread appends the Transaction structure into the “transactions” linked list which is also done by locking the mutex “transaction_mutex”. After that, seller thread frees its transaction in the corresponding Seller_Request struct and signals the condition variable to wake up its customer. Then it waits for a new transaction from a customer.

The synchronization of the threads between day changes are done using a condition variable. When the day is over, main thread changes dayEnd variable to 1. Then it waits other threads to quit their job and sleep. When the seller threads detect dayEnd, they try to lock themselves. If there is a customer that has the lock, seller signals it continuously until customer unlocks and then the seller gets its lock. When the threads detect dayEnd, they decrement the variable “threadsReady” which was initially set to the sum of number of seller and customer threads. When the threadsReady becomes 0 it means that every thread has detected dayEnd and then the main thread initializes the product counts, and signals all

the threads waiting for the condition variable with pthread_cond_broadcast() function. Finally it increments the currentday variable.