Salwa Haider

CS 4348.001

Bank Simulation

1. A list of every semaphore, its purpose, and its initial value
2. tellerReady [] – array of semaphores to keep teller threads waiting until a customer is ready and has been waiting in the teller line. Initial value is 0. When a customer is ready, teller threads unblock and execute.
3. loanOfficerReady [] – array of semaphores to keep loan officer thread waiting until a customer is ready and waiting in loan officer line. Initial value is 0 and when customer signals they are ready, loan officer threads unblock and continue execution.
4. waitTellerLine – mutual exclusion for queue to represent the line for bank teller so that adding or removing from queue does not change multiple threads.
5. waitOfficerLine – mutual exclusion for queue to represent the line for loan officer so that adding or removing from queue does not change multiple threads.
6. TellerQueueNotEmpty – semaphore to represent when the teller queue is full. Initial value is 0.
7. OfficerQueueNotEmpty – semaphore to represent when the officer queue is full. Initial value is 0.
8. RequestBankTeller [] – an array of semaphores for each bank teller used to stop teller threads until customer signals the teller that a request was made.
9. depositReceipt [] – an array of semaphore to represent that a receipt has been deposited.
10. depositComplete [] – an array of semaphore to represent deposit being complete.
11. withdrawReceipt [] – array of semaphore to represent withdrawal receipt.
12. withdrawalComplete [] – array of semaphore to represent withdrawal being complete.
13. loanTransactionComplete – semaphore to represent loan transaction has been completed.
14. loanOfficerRequest – a semaphore for loan officer used to stop the officer thread until customer signals the officer that a request was made.
15. loanOfficerReceipt – semaphore to represent loan officer receipt.
16. Pseudocode for each function

Customer()

for(i = 0; i < 3; i++)

{

enterBank(); //create customer thread

assigntask(); //randomly assign task to customer

taskCustomer[amount]; //random amount assigned for the task

wait(waitTellerLine); //customer added to queue using mutex

TellerQueue(); //customer waiting in queue

signal(waitTellerLine); //done with mutex

signal(tellerReady); //customer signals to be ready and waiting in line

wait(taskCustomer[i]); //customer waits for employee to be assigned

requestTask(); //customer requests task

signal(request); //customer signals that he or she made a request

wait(performTask[i]); //customer waits for task to be completed

getReceipt(); //get receipt from teller or officer

visit++; //visit again

}

Main()

{

createSemaphores(); //initialize semaphores

createThreads(); //create threads

startThreads(); //run each thread

joinThreads(); //join threads

printSummary(); //print summary table

}

Bankteller()

while(true)

{

wait(TellerQueueNotEmpty); //wait till bank teller queue is not empty

wait(waitTellerLine); //access queue

getCustomerFromLine(); //getting customer from line

signal(waitTellerLine); //done using queue

signal(tellerReady[i]); //signal customer that they have been assigned to teller

wait(request); //wait for customer request

processLoan(); //loan is processed

signal(performTask[i]); //signal customer that task has been done

}

LoanOfficer()

while(true)

{

wait(OfficerQueueNotEmpty); //wait till loan officer queue is not empty

wait(waitOfficerLine); //access queue

getCustomerFromLine(); //getting customer from line

signal(waitOfficerLine); //done using queue

signal(officerReady[i]); //signal customer that they have been assigned to officer

wait(request); //wait for customer request

processTransaction(); //transaction is processed

signal(performTask[i]); //signal customer that task has been done

}