Semaphores:

* cust\_ready\_teller
  + Determines whether a customer is ready (in the teller line).
  + Initial value = 0
* cust\_ready\_loan
  + Determines whether a customer is ready (in the loan line).
  + Initial value = 0
* teller\_ready
  + Determines whether a teller is open.
  + Initial value = 2
* loanofficer\_ready
  + Determines whether a loan officer is open.
  + Initial value = 1
* access\_balance
  + Enforces mutual exclusion of modification of balance.
  + Initial value = 1
* access\_loan\_amt
  + Enforces mutual exclusion of modification of the loan amount.
  + Initial value = 1
* queue\_mutex
  + Enforces mutual exclusion of adding/removing from the customer queue.
  + Initial value = 1
* loan\_finished[5]
  + Notify the customer that the loan officer has finished
  + Initial value = 0
* teller\_finished[5]
  + Notify the customer that the teller has finished
  + Initial value = 0

Functions:

**class Customer**

Integer balance

Integer thread\_number

Integer num\_times\_visited

void run()

{

times\_visited\_bank[cust\_number]++

if( times\_visited\_bank == 3)

return

signal(cust\_ready\_teller)

wait(teller\_ready)

choice = rand(0 or 1 or 2)

amount = rand(100 or 200 or 300 or 400 or 500)

wait(queue\_mutex)

customer\_queue.push()

signal(queue\_mutex)

}

**class BankTeller**

Integer thread\_number

void run()

{

wait(queue\_mutex)

customer\_queue.pop()

signal(queue\_mutex)

wait(cust\_ready\_teller)

if(choice = 0)

deposit(amount)

if(choice = 1)

withdraw(amount)

signal(teller\_ready)

}

Integer deposit(Integer amount)

{

wait(access\_balance)

balance = balance + amount

signal(access\_balance)

}

Integer withdraw(Integer amount)

{

wait(access\_balance)

balance = balance – amount

signal(access\_balance)

}

**class LoanOfficer**

void run()

{

wait(cust\_ready\_teller)

loan(amount)

signal(loanofficer\_ready)

}

Integer loan(Integer amount)

{

wait(access\_loan\_amt)

loan\_amt = loan\_amt + amount

signal(access\_loan\_amt)

}

**class Main**

Customer customer\_queue

Semaphore queue\_mutex

Semaphore access\_loan\_amt

Semaphore access\_balance

Semaphore loanofficer\_ready

Semaphore teller\_ready

Semaphore cust\_ready\_teller

Semaphore cust\_ready\_loan

Semaphore teller\_finished[5]

Semaphore loan\_finished[5]

Main()

{

Thread loan\_officer

Thread teller\_1

Thread teller\_2

Customer [5] customers

for(customer in customers)

customer.start()

teller\_1.start()

teller\_2.start()

loan\_officer.start()

}