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()
}
```