//program design

//my semaphores, purpose, and initial values

semaphore customer = 5; //limit the amount of customers to 5

semaphore bank\_teller = 2; //limit the amount of customers that can access bankteller to 2

semaphore loan\_officer = 1; //limit the amount of customers that can access loanofficer to 1

semaphore cust\_ready1 = 0, cust\_ready2 = 0 finished\_bank = 0, finished\_loan = 0, tellerprint1 = 0, loanprint1 = 0, custreqteller = 0, custreqloan = 0;

/\* (in order)

used to signal the bankteller that customer is ready.

used to signal the loan officer that customer is ready.

used to signal customer that bankteller has finished processing a customer

used to signal customer that loanofficer has finished processing a customer

used for customer to wait for teller to print that it is helping

used for customer to wait for loan officer to print that it is helping

used to signal teller that customer has finished making a request

used to signal loan officer that customer has finished making a request

\*/

semaphore mutex1 = 1; //use when enqueueing in to loan officer

semaphore mutex2 = 1; // use when enqueueing in to bank teller

semaphore mutex3 = 1; //use when teller is dequeueing

semaphore mutex4 = 1; // use when loan officer is dequeueing

semaphore mutex4 = 1; // use for the for loop in customer

//pseudocode for each function

void\* customer(void \* arg){

int \*customer\_num = (int \*) arg;

wait(customer);

//loop to let each customer make 3 visits to bank

wait(mutex5);

for(int x=0; x<3; x++){

int task = getrandomtask(); //set task to number 1 or 0

if(task == 0){ //go to bank teller

wait(bank\_teller);

wait(mutex2);

enqueue(\*customer\_num);

signal(cust\_ready1);

signal(mutex2);

wait(tellerprint1);

signal(custreqteller);

wait(finished\_bank);

}

if(task == 1){ //go to loan officer

wait(loan\_officer);

wait(mutex1);

enqueue(\*customer\_num);

signal(cust\_ready2);

signal(mutex1);

wait(loanprint1);

signal(custreqloan);

wait(finished\_loan);

}

}//end for loop

signal(mutex5);

exit\_bank();

signal(customer);

}

void\* bankteller(void \* arg){

int \*tellernum = (int\*) arg;

while(true)

{

wait(cust\_ready1);

wait(mutex3);

dequeue(\*tellernum);

signal(tellerprint1);

wait(custreqteller);

bankservecust();

signal(mutex3);

signal(finished\_bank);

signal(bank\_teller);

}

}

void\* loanofficer(void \* arg){

while(true)

{

wait(cust\_ready2);

wait(mutex4);

dequeue();

signal(loanprint1);

wait(custreqloan);

loanservecust();

signal(mutex4);

signal(finished\_loan);

signal(loan\_officer);

}

}

main(){

initialize semaphores

create threads for customer

create threads for bank teller

create threads for loan officer

join threads for customer

destroy the semaphores

print end values for balance and loan amount

}