**List of Semaphores**

* box\_agent
  + initial value: 2
  + used to allow only 2 people to have access to the box office agent at once
* concession\_worker
  + initial value: 1
  + used to allow only 1 person to have access to the concession stand at once
* give\_ticket
  + initial value: 0
  + used by the box office agent to tell the customer that they received their ticket and can continue
* hand\_over\_ticket
  + initial value: 0
  + used by customer to tell the ticket taker to start tearing the ticket
* mutex0
  + initial value: 1
  + used by box office agent to give them mutual exclusion over the agent\_count variable (for agentnr)
* mutex1
  + intial value: 1
  + used by customer to have mutual exclusion over cust\_count variable (for custnr)
* mutex2
  + initial value: 1
  + used by both customer and box office agent to have mutual exclusion over agent queue
* mutex3
  + initial value: 1
  + used by both customer and ticket taker to have mutual exclusion over taker queue
* mutex4
  + initial value: 1
  + used by both customer and concession worker to have mutual exclusion over conc queue
* mutex5
  + initial value: 1
  + used by printmsg() to provide mutual exclusion for output in order to prevent output corruption
* order\_filled
  + initial value: 0
  + used by concession worker to tell customer that the order is filled and can continue
* order\_ready
  + initial value: 0
  + used by customer to tell concession worker to work on his order
* ticket\_payment
  + initial value: 0
  + used by customer to tell box office agent to sell him a ticket
* ticket\_taker
  + initial value: 1
  + used to allow only 1 person to have access to the ticket taker at once
* ticket\_torn
  + initial value: 0
  + used by ticket taker to tell customer that the ticket has been torn and can continue

Pseudo-code:

/\* program theater \*/

int total\_customer = 300, total\_box\_agent = 2, total\_concession\_worker = 1, total\_ticket\_taker = 1;

semaphore box\_agent = total\_box\_agent, concession\_worker = total\_concession\_worker, ticket\_taker = total\_ticket\_taker;

semaphore give\_ticket = 0, hand\_over\_ticket = 0, order\_filled = 0, ticket\_torn = 0, order\_ready = 0, ticket\_payment = 0;

semaphore mutex0 = 1, mutex1 = 1, mutex2 = 1, mutex3 = 1, mutex4 = 1, mutex5 = 1;

int time\_tear\_ticket = 15, time\_conc\_order = 180, time\_sell\_ticket = 90;

bool ticket[total\_customer] = {false} //each bool refers to a customer's ticket status, false = no ticket, true = received ticket

int movie[total\_customer] = {0} //each int refers to a customer's movie choice

string c\_order[total\_customer] = {""} //each int refers to a customer's order at the concession stand, empty string = no order ()

int cust\_count = 0, agent\_count = 0;

void main(){

count = 0;

read\_in\_file()

for i=0 to total\_customer-1{

create thread\_customer()[i];

}

for 1 to total\_box\_agent{

create thread\_box\_agent();

}

for 1 to total\_concession\_worker{

create thread\_concession\_worker();

}

for 1 to total\_ticket\_taker{

create thread\_ticket\_taker();

}

for i=0 to total\_customer-1{

thread\_customer[1].join();

}

}

void thread\_customer(){

int custnr;

wait(mutex1); -297 [cust3, 4, 5, ....]

cust\_count++;

custnr=cust\_count;

signal(mutex1);

announce\_customer\_created();

pick\_movie();

wait(box\_agent);

wait(mutex2);

enqueue\_agent(custnr);

signal(ticket\_payment);

signal(mutex2);

wait(give\_ticket);

signal(box\_agent);

if ticket[custnr] == true {

ticket\_recieved();

get\_in\_t\_line();

wait(ticket\_taker);

wait(mutex3);

enqueue\_taker(custnr);

give\_to\_taker();

signal(hand\_over\_ticket);

signal(mutex3);

wait(ticket\_torn);

signal(ticket\_taker);

if rand > 50%{

c\_order[custnr] = pick\_concession();

get\_in\_c\_line();

wait(concession\_worker);

wait(mutex4);

enqueue\_conc(custnr);

placing\_order();

signal(order\_ready);

signal(mutex4);

wait(order\_filled);

receive\_order();

signal(concession\_worker);

}

enter\_movie();

}

else

leave\_theater();

}

void thread\_box\_agent(){

int agentnr;

int b\_cust;

wait(mutex0);

agent\_count++;

agentnr = agent\_count;

signal(mutex0);

announce\_agent\_created();

while(true){

wait(ticket\_payment);

wait(mutex2);

dequeue\_agent(b\_cust);

if movie[b\_cust] is available

ticket[b\_cust] = true;

signal(mutex2);

delay(time\_sell\_ticket);

signal(give\_ticket);

}

}

void thread\_ticket\_taker(){

int t\_cust;

announce\_taker\_created();

while(true){

wait(hand\_over\_ticket);

wait(mutex3);

dequeue\_taker(t\_cust);

signal(mutex3);

delay(time\_tear\_ticket);

ticket\_taken();

signal(ticket\_torn);

}

}

void thead\_concession\_worker(){

int c\_cust;

announce\_conc\_created();

while(true){

wait(order\_ready);

wait(mutex4);

dequeue\_conc(c\_cust);

signal(mutex4);

order\_taken();

delay(time\_conc\_order);

give\_order();

signal(order\_filled);

}

}

void printmsg(string s){ //used to keep output from being corrupted

wait(mutex5);

print(s);

signal(mutex5);

}