****

**De La Salle University • College of Computer Studies**

**Synchronization Problems**

Name (last name first) : Fernandez, Ryan Austin

Poblete, Clarisse Felicia M.

San Pedro, Marc Dominic

Tan, Johansson E.

Section : G01

Date of Submission : February 23, 2016

**Sleeping Barber**

**Semaphores**

Semaphore x = 1, barber = 0, waitBarber = 1;

boolean barbering = false;

int queue = 0,

MAX\_QUEUE = 4;

Customer() {

while(true) {

boolean ok = true;

x.wait();

if(barbering) {

if(queue < MAX\_QUEUE) {

queue++;

x.signal();

waitBarber.wait();

x.wait();

queue--;

barber.signal();

barbering = true;

ok = true;

} else {

ok = false;

}

}

x.signal();

if (ok) {

getHaircut();

} else {

leave();

}

}

}

Barber() {

while(true) {

barber.wait();

giveHaircut();

waitBarber.signal()

}

}

**Monitors**

Monitor SB {

int queue = 0;

int MAX\_QUEUE = 4;

boolean barbering = false;

boolean heyBarber = false;

Condition waitBarber;

boolean checkBarber() {

while(barbering) {

if(queue < MAX\_QUEUE) {

queue++;

waitBarber.wait();

queue--;

} else {

return false;

}

}

barber.notify();

heyBarber = true;

barbering = true;

return true;

}

barberSleep() {

if( heyBarber) {

heyBarber = false;

} else {

barber.wait();

}

}

finishCut() {

barbering = false;

waitBarber.signal();

}

}

SB sb;

Barber() {

while(true) {

sb.barberSleep();

giveHaircut();

sb.finishCut();

}

}

Customer() {

while(true) {

if (sb.checkBarber()) {

getHaircut();

} else {

leave();

}

}

}

**Roller Coaster**

**Semaphore**

Semaphore x = 1, y = 1, full = 0, empty = 0,boardCond = 0, leaveCond = 0;

int trainCap = 0;

boolean hasWaiting = false;

Passenger() {

while(true) {

x.wait();

boardCond.wait();

trainCap++;

if( trainCap == CAPACITY ) {

full.signal();

}

x.signal();

board();

y.wait();

leaveCond.wait();

trainCap--;

if( trainCap == 0) {

empty.signal();

}

y.signal();

leave();

}

}

Train() {

while(true) {

load();

for(int i = 0; i < CAPACITY; i++) {

boardCond.signal();

}

if(trainCap < CAPACITY) {

full.wait();

}

run();

unload();

for(int i = 0; i < CAPACITY; i++ ) {

leaveCond.signal();

}

if(trainCap > 0) {

empty.wait();

}

}

}

**Monitors**

c = capacity

Monitor m{

Cond loadQ,

runQ,

unloadQ,

endUnloadQ

loaded = 0

loading = false

unloading = false

tryBoard(){

while(!loading){

loadQ.wait()

}

loaded++

if(loaded == c){

loading = false

runQ.notify()

}

}

startLoad(){

loading = true

loadQ.notifyAll()

}

tryRun(){

while(loaded < c){

runQ.wait()

}

}

startUnload(){

unloading = true

unloadQ.notifyAll()

while(unloading){

endUnloadQ.wait()

}

}

tryLeave(){

while(!unloading){

unloadQ.wait()

}

loaded--

if(loaded == 0){

unloading = false

endUnloadQ.notify()

}

}

}

Passenger (multiple) - - - - - - - - - - - - -

m.tryBoard()

board() --> Only if the car calls load

m.tryLeave()

leave() --> Only if the car calls unload

Car (single) - - - - - - - - - - - - - - - - -

while(1){

load()

m.startLoad()

m.tryRun()

run() --> Only if the car is full

unload()

m.startUnload()

}