/\*program postoffice\*/

// 10 customers can be inside post office at a time

**semaphore** max\_capacity = 10;

//1 customer or post worker can modify the queue at a time

**semaphore** mutex = 1;

//1 post worker can increment count at a time

**semaphore** mutex2 = 1;

//post worker should wait until customer signals customer\_ready

**semaphore** customer\_ready = 0;

//1 post worker can use the scale at a time

**semaphore** scale = 1;

int count = 0; //count the number of customer served

//customer\_node data structure

struct customer\_node{

int customer\_number;

int task = assign\_task(); //get random number between 0,1,2

//customer may leave after post worker signals finished

**semaphore** finished = 0;

customer\_node\* next = NULL;

}

//initialize queue to empty

customer\_node\* head = NULL;

customer\_node\* tail = NULL;

//function definition serve\_customer

void serve\_customer(customer\_node\* cust, int worker\_number){

switch(cust->task){

case 0:

buy\_stamps();

case 1:

mail\_letter();

case 2:

mail\_package();

semWait(scale);

use\_scale();

semSignal(scale);

}

}

void customer(customer\_node\* cust){

semWait(max\_capacity);

enter\_post\_office();

semwait(mutex);

enqueue (cust, head, tail);

semSignal(mutex);

semSignal(customer\_ready);

semWait(cust->finished);

leave\_post\_office();

semSignal(max\_capacity);

}

void post\_worker(int worker\_number){

customer\_node\* cust;

while(true){

semWait(customer\_ready);

if(count == 50) // if all customers have been served, exit

break;

semWait(mutex);

cust = dequeue(head, tail);

semSignal(mutex);

serve\_customer(cust, worker\_number);

semSignal(cust->finished);

semwait(mutex2);

count++;

semSignal(mutex2);

if(count == 50){

for(int i = 0; i < 2; ++i) //signal the other two workers to exit

semSignal(customer\_ready);

break;

}

}

}

void main(){

for(cust\_number = 0; cust\_number < 50; cust\_number++){

create new customer\_node\* cust;

cust->customer\_number = cust\_number;

create\_thread(customer, cust);

}

for(worker\_number = 0; worker \_number < 3; worker \_number++){

create\_thread(post\_worker, worker\_number);

}

for(cust\_number = 0; cust\_number < 50; cust\_number++){

join\_thread(customer, cust\_number);

}

for(worker \_number = 0; worker \_number < 50; worker \_number++){

join\_thread(post\_worker, worker\_number);

}

}