**Project Design**

**Semaphores**

ArrayList<Semaphore> cleft, cleft[i] = 0 where i ranged from 0-49 //customers left post office

**Semaphore** capacity = 10 //at most 10 customers are allowed in the post office

**Semaphore** wcapability = 3 //3 workers can provide service

**Semaphore** cready = 0 //customer is ready

**Semaphore** staticMutex = 1 // customer and postal worker tied to each other to protect static variables

**Semaphore** c\_w = 0 //mutex that ensures customer requests assistance at appropriate time

**Semaphore** request = 0 //customer service request

**Semaphore** scale = 1 //only one worker can use the scale

**Semaphore** working = 0 //postal worker is working or not

**Pseudocode:**

**void** Postal\_Worker

{

wait (cready) //gets a customer that can be served

get statistic values which are a customer (with number 0-49) and service type (buy stamps, mail a package, or mail a letter)

and assign this worker to the that customer

signal (c\_w) // postal worker gets static data

wait(request) //wait permission to start service

//check which type of service is requested by that customer

if mailing a package

wait(scale) // wait until scale is available for this worker

wait until service done

else // buy stamps or mail a letter

wait until service done

signal (cleft (customer number)) // worker finish service to this customer

wait (working) //waits for customer to exit, then this worker is free to service next customer

signal (wcapability) //allows another customer to be helped

}

**void** Customer

{

wait (capacity) //customer enter post office

wait (wcapability) //customer get one postal worker

wait (staticMutex) //customer and postal worker tied to each other, only this thread can write to static variables

get statistic values which are a customer (with number 0-49) and service type (buy stamps, mail a package, or mail a letter)

signal (cready) //allows worker to begin reading data

wait(c\_w) //wait for worker to acquire static data

signal(request) //allows worker to begin working on service

signal(staticMutex) //other threads can now write to static variables

wait (cleft (customer number)) // wait until worker finish service to this customer

signal (working) //this customer leaves and worker is not working now

signal (capacity) //then allows one more customer to enter the post office

**void** Post Office

{

// static variables could pass to customer and postal worker class

static int customer\_number, service\_type

static int [50] assign\_worker

**void main**

{

Create 50 customers and corresponding threads, start threads

Create 3 workers and corresponding thread, start threads

Joint customers into original process once they are done

}

}