Anupama Vijayan
Netid: axv146030

DESIGN

LIST OF SEMAPHORES:

SEMAPHORE	INITIAL VALUE	PURPOSE
max_capacity	10	to have only 10 customers to enter the post office at a time
worker	3	To have the 3 workers to serve the customers
free_counter	0	To let the worker know that the customer has left and the counter is free
counter_available	0	To signal that the worker is free to serve the next customer
cust_ready	0	To signal the worker that the customer is ready to be served
mutex1	3	It allows only 3 customers to go to counters, as there are only 3 workers available in the post office
mutex2	1	To ensure mutual exclusion is there for the workers
mutex3	1	To let only 1 worker to access the Scale at a time
customer_entered	0	To ensure that the worker starts serving only after the customer comes to the counter
serve_signal	1	To let the customer know that he can ask for the service he wants
order_finish	0	To signal that the customer has confirmed that he is done with his work
order	0	To let the worker wait for the customer to order what service he wants
finished[]	0	An array of semaphores used to signal the customer that the worker is done

PSEUDOCODE:

```
int TotalCustomers=50;
int TotalPostalWorkers=3;
int cust_index;
Semaphore max_capacity = 10 ;
Semaphore free_counter =0;
Semaphore counter_available = 0;
```

```
 Semaphore worker= 3;
 Semaphore cust_ready=  0;
 Semaphore mutex1=  3;
 Semaphore mutex2=  1;
 Semaphore mutex3=  1;
 Semaphore customer_entered=  0;
 Semaphore serve_signal=  1;
 Semaphore order_finish=  0;
 Semaphore order=  0;
 Semaphore finished[50] = {0};

PostOffice()
{
Create 50 threads for  customers
Create 3 threads for worker
}
Customer()
{
wait(max_capacity);
enters_PostOffice()
wait(serve_signal);
wait(mutex1);
go_to_counter();
signal(cust_ready);
signal(customer_entered)
wait(counter_available);
customer_orders_service();
signal(order);
wait(finished[cust_index]);
wait(order_finish);
signal(mutex1);
leaves_office();
signal(free_counter);
signal(max_capacity);
}

PostalWorker()
{
wait(cust_ready);
wait(worker);
wait(customer_entered);
wait(mutex2);
signal(serve_signal);
signal(counter_available);
wait(order);
signal(mutex2);
serves_customer_order();
signal(finished[cust_index]);
```

```
signal(order_finish);
wait(free_counter);
signal(worker);
}
```
