Homework 2 for CS153 (Fall 20 1 9)

Due: 11/3/2019

Instructions:

I. Write synchronization code to simulate each of the following scenarios:

a. (3 points) A barrier: a group of us go to a restaurant; we wait until the last person arrives before we go in.

```
Semaphore barrier(0); // set the varrier to 0 otherwise is 1
int people = 10; //Assume ther eare 10 people for the resturant
int arrived = 0;
arrive(){
mutext.wait();
people--;
arrived++;
if (people == 0)
{
   while(arrived == 10)
   {
        barrier.signal(); // send 10 people arrived signal to console
   }
}
mutex.signal();
barrier.wait();
}
```

b. (3 points) A bakery where threads of three types representing three ingredients cake mix, filling and icing arrive. Whenever we have one of each, we make a cake.

```
Semaphore mutex(1), icing(1),cake(1),mix(1);
int icing = 0, filling = 0, mix = 0;
bool cakeDone = flase;
AddIcing(){
        icing.wait();
        mutex.wait();
        if(mix == 1 && filling ==1 && mix ==1){
        cakeDone = true;
        icing.signal();
        filling.signal();
        mix.signal();
        }
        else{
        icing++;
        }
mutex.signal();
}
```

c. (3 points) The recipe in part b has changed – now we need two portions of cake mix to arrive (in addition to filling and icing) before we can make cake. Update your implementation.

```
Semaphore mutex(1), icing(1),cake(1),mix(1);
int icing = 0, filling = 0, mix = 0;
bool cakeDone = flase;
AddIcing(){
        icing.wait();
        mutex.wait();
        if(mix == 1 && filling ==1 ||  mix ==1 && icing ==1 || icing ==1 && filling ==1 ){
        cakeDone = true;
        icing.signal();
        filling.signal();
        mix.signal();
        }
        else{
        icing++;
        }
mutex.signal();
}
```

II. You are writing code for the voting machines for an upcoming election. You use shared counters, one for each candidate to keep track of the votes as they come from the different voting machines. You can think of each machine as a thread: every time it receives a vote, it increments a counter for that candidate.

a. (2 point) Explain what could go wrong with this implementation if we do not use synchronization

We will lose the collection of tickets from voters. Because they are using multiple machines to submit their tickets at the same time.

b. (2 points) Suggest two ways to use locks to solve this problem without changing the code other than adding the lock operations; which one is more conservative (i.e., more restrictive).

We can design a voter counter lock, or we can design for each candidate a ticket lock. The first one is more general, because each voter can only vote once. The second one is more flexible, it allows one voter select two different candidates, which we might not need to see it happens.

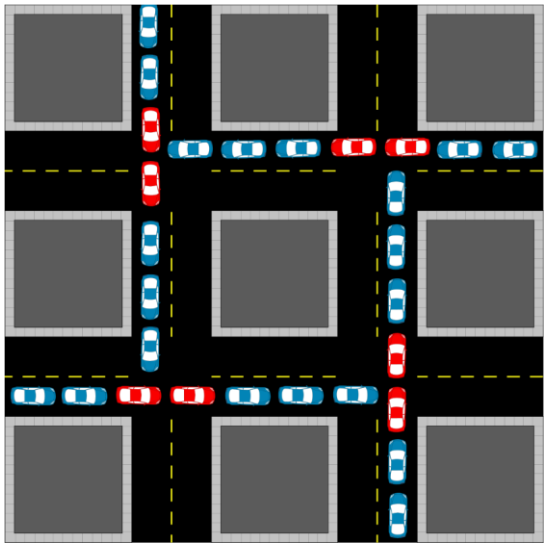
c. (1 point) Consider the following improvement to the implementation suggested by a cs153 veteran: for each thread, maintain a local count of the votes, and then update the global count periodically. Do we still need synchronization?

We don’t need synchronization, we need privatization instead. Because the internal private ticket data is invisible from outer global ticket data. The privatization will be good to optimize periodically programs.

d. (1 point) Compare the implementation in c to the better of the two implementations in b.

The implementation in c reduced the synchronization process in implementations in b. Also reduced the system recourses usage. And we will not need to consider the lock is dead-lock or some other conditions. Also the global and private variable are very easy to implement and less error will encounter.

III. (5 points) Traffic in Manhattan goes around a block as shown in the figure below.



Picture from: Wikipedia (https://upload.wikimedia.org/wikipedia/commons/thumb/d/d9/Gridlock.svg/440px-Gridlock.svg.png)

Having studied concurrency, you recognize that even though we call this gridlock, this situation may be a case of deadlock. Use our criteria for deadlock to show whether this is indeed deadlock or not.

If this is deadlock, discuss and compare two solutions to prevent it from happening.

The intersection has 2 ways of traffic and each way doesn't interrupt others so has the quality of mutual exclusion. And the traffic lights will represent the hold and wait concept of the deadlock. Also, the traffic is one way sided. You cannot drive your car in a reverse way, so it qualifies the identity of preemption of deadlock. And from the graph you can see each way traffic is waiting for another side traffic. Only one direction traffic moves then the next traffic intersection will move. So, it has the circular concept of the deadlock. Therefore this is a deadlock.

How to prevent it happened?

1. We can make traffic of two sides goes together. For example. The traffic from North to South and the traffic from South to North can go together. Also, the traffic from East to West and the traffic from West to East can go together.
2. Adding do not block the intersection sign to the cross-road will help to clear the intersection.