2 c) Why is it not advisable to set the value of X to be too small (e.g., = 1)?

It is not advisable to set the vale of X to a small value because it would reduce efficiency greatly. If, for example, X = 1, then eventually there would come a time where north and south bound cars would alternate taking turns on the strip of highway, 1 car at a time. This would prevent cars traveling concurrently on the highway, which would greatly slow increase the average time it takes to cross this section of highway.

3) Write pseudo-code for the CS students as well as that for their friends in Political Science.

```
int lastfriend = 0;            // Last PS friend to be woken up
Semaphore mutex = 1;
Semaphore preparing = 0;
Semaphore order = 0;

CS_student(){
    repeat forever{
        // Student must wait for their turn
        wait(mutex);

        // Check if coffee pot is empty. If so, order more
        if (coffeepot == 0){
            signal(order);

            // Wake up a Political Science friend, and wait for coffee
            revive( Friend [((lastfriend + 1) % F)];
            wait(preparing);
        }

        // Fill mug
        coffeepot--;
        signal(mutex);
        study and drink coffee until mug is empty
    }
}

PoliSci_Friend(){
    repeat forever{
        Do Stuff();            // Do some Political Science hw, perhaps?

        wait(order);           // Wait for an order to be placed
        coffeepot = M;         // Fill pot with M mugs of coffee
        signal(preparing);     // Order is done!
        suspend();             // Sleep until woken up
    }
}
```

4) This pseudo-code assumes that when a customer arrives, the employees of KMSO.com already in the bathroom are able to finish using the bathroom.

```
Semaphore mutex = 1;
Semaphore lock = 1;                 // lock for bathroom door
Semaphore KMSOcount = 4;            // Allows 4 KMSO employees at a time

int customer = 0;                   // Number of customers waiting + in bathroom
int waiting = 0;                    // Number of people waiting
int occupancy = 0;                  // Number of people in bathroom

employee(){
        repeat forever{

                do work();

                // Entry protocol
                wait(mutex);
                waiting++;          // Actually in the bathroom at this point
                signal(mutex);

                wait(KMSOcount);    // Wait until there are less than 4 employees
                while(customer > 0){}       // Wait until there are no customers

                wait(mutex);
                occupancy++;        // Actually in the bathroom at this point
                waiting--;
                signal(mutex);

                // If you are first, lock the door (now only employees are allowed in)
                if( occupancy == 1) { wait(lock); }

                if (lights == off) { lights = on; }     // Make sure lights are on
                use_bathroom();                         // Use bathroom

                wait(mutex);
                if ((occupancy == 1) && (waiting == 0) { lights = off; }
                occupancy--;
                signal(mutex);

                signal(KMSOcount);                      // Tell other employees there is room
                if(occupancy == 0){ signal(lock); }     // Allow anyone in next
        }
}
```

```
customer(){

    repeat forever{

        shop_at_Dunkin();

        // Entry protocol for using bathroom
        wait(mutex);
        customer++;          // This will make all employees not yet in bathroom wait
        waiting++;
        signal(mutex);

        wait(lock);          // Wait for all employees to vacate the bathroom

        wait(mutex);
        waiting--;           // Now we are in the bathroom
        signal(mutex);

        if(lights == off) { lights = on; }       // Turn lights on if needed
        use_bathroom();

        if(waiting == 0) { lights = off; }       // Turn lights off if no one else is waiting

        wait(mutex);
        customer--;                              // Leaving the bathroom
        signal(mutex);

        signal(lock);                            // Unlock door for next person
    }
}
```

5) What is the minimum values of X, Y, Z that would render the state below a safe state?

Claim Matrix - Allocation Matrix = Need Vector

	R1	R2	R3
P1	3	1	4
P2	6	1	3
P3	3	2	2
P4	4	2	2

	R1	R2	R3
P1	2	1	1
P2	5	1	1
P3	2	0	1
P4	0	0	2

	R1	R2	R3
P1	1	0	3
P2	1	0	2
P3	1	2	1
P4	4	2	0

If we have X = 1, Y = 0, Z = 2, the above will be a safe state:

We can allocate (1, 0, 2) to P2. Upon completion, we have (6,1,3) available.

Next, we allocate (1,0,3) to P1. When completed, we have now (8,2,4) available.

Next, give (1,2,1) to P3. When finished we have (10,2, 6) available.

Finally, give (4,2,0) to P4, leaving us with (10,2,8).

Since there is never a time when all requests are greater than the available resources (meaning at least one process can finish) this state is safe.