Writing Assignment Solutions:
#1----------
Proof by contradiction:
    1. Assume P0 and P1 are in their critical sections.
    2. flag[0] and flag[1] cannot be true at the same time.
    3. P0 allows P1 access with the turn variable and vice versa
    4. Assume P0 had enetered the critical section then turn would need to be 1
    5. At this point P1 would never get access to critical section because in order
        for turn to be 0 P0 would need to not be in the critical section.
    6. P0 cannot be inside the critical and outside of the critical section at the
same time

#2----------
Proof by contradiciton:
    NOTE::PN means a random process from P0 to PK

    1. Assume P0 through PK are in the critical section
    2. PN has the lowest number or the same number with a lower pid
    3. Every process after will recieve a higher number than PN at least
    4. Which means in order for another process to get in the critical section
        it would need to get a lower number than PN which is not possible
        since number[i] = max(number) + 1
    5. Basically if process P0 and P1 are in the critical section then
        P0's number < P1's number and P1's number < P0's number which is not
possible

#3----------
```
//Solving probelm using binary semaphores
writers_semaphore = 1;
readers_semaphore = 1;

//number of readers in critical section
readers = 0;

//basically just need to check if we can access to resource because readers
//do most of the logic for us
Writers Process:
    wait(writers_semaphore);
    << write to critical section >>
    signal(writers_semaphore);

//this is the most complex process as it allows the writer access or not
Readers Process:
    wait(readers_semaphore);

    //check if we need to lock writers out of critical section
    //if we are the first reader lock writers out of critical section
    if(readers == 0)
        signal(writers_semaphore);

    readers++;

    signal(readers_semaphore);

    //we have now locked out all writers
    << read from critical section >>
```

```
    //we are not reading so lock and subtract from readers
    wait(readers_semaphore);

    readers--;

    //checking if we should allow writers back in critical section
    //if readers is 0 then writers can have access
    if(readers == 0)
        signal(writers_semaphore);

    signal(readers_semaphore);
-------------------------------------------

#4---------------------------------------
Shared Data:
blocked: array[0..1] of Boolean;
turn: 0..1;

blocked[0] = blocked[1] = false;
turn = 0;

Local Data:
ID: 0..1;  /* (identifies the prccess;
              set to 0 for one process,
              1 for the other) */

ORIGINAL CODE:
while(1){
    blocked[ID] = true;
    while(turn <> ID){
        while(blocked[1 -ID]);
        turn = ID;
    }

    << critical section >>

    blocked[ID] = false;

    << normal work >>
}

WHAT RUNS ON EACH PROCESS TO ENTER CRIT SECTION:
P0                                  |P1
                                    |   while (1) {
                                    |       blocked[ID] = true;
                                    |       while (turn <> ID){
                                    |           while (blocked[1 – ID]);
while (1) {                         |
    blocked[ID] = true;             |
    while (turn <> ID)              |
                                    |           turn = ID;
                                    |      }
                                    |      while (turn <> ID)

    ----------------BOTH ARE IN THE CRITICAL SECTION----------------
    << critical section >>          |       << critical section >>
-------------------------------------------

In the begging turn = 0 and blocked[0] = blocked[1] = false.
```

```
//P1 running
P1 starts running and sets blocked[1] = true.
P1 checks if turn != ID which is true so it enters the while loop.
P1 then checks if blocked[1-1] which is blocked [0] which is false.

//P2 running
P2 then starts running and sets blocked[0] = true;
P2 checks if turn != ID which is false so it continoues to the critical section.

//P1 running
P1 then sets turn to ID which is setting turn to 1.
P1 then starts over the while loop and checks if turn != ID which is false
P1 then enters the critical and section and both processes are in the critical
section.
-------------------------------------------
```