Andrew Candelaresi
Problem set 2

1.
```
1. eastcount =0
   westcount = 0
   westlock =1
   mutex = 1
   mutex2 =1
//going east
 while(1) {
            wait(mutex);
            eastcount++
            if  (eastcount==1) wait(westlock);
            signal(mutex);
            //go east
            wait(mutex);
            eastcount--;
            if (eastcount==0) signal(westlock);
...         wait(mutex)
}
//going west
while(1){
            wait(westlock);
            wait(mutex2);
            westcount++;
            if (westcount==1) wait(easlock);
            signal(mutex2);
            //go west
            wait(mustex2);
            westcount--;
            if(westcount ==0) signal(eastlock);
            signal(mutex2);
...
}
```
cars going east will be allowed to go first and could starve out cars going west if the east count is never decremented to zero.

2.
```
lock mutex;
condition x;
int P1_complete =0;

//P1 starts
aquire(mutex);
//execute code C1
P1_complete = 1;
 release(mutex);
c.signal()
```

```
//end P1

//P2
Acquire(mutex);
while (!P1_complete) {
   Release(mutex);
   c.wait();
   Acquire(mutex);
}
// execute code c2
Release(mutex);
// End P2
```

3. Because it creates a condition where readers are forced to queue outside of readblock when a writer is waiting and so they no longer enter the mutex that allows them to increment the readcount. This ensures that eventually the readcount will be decremented to zero which will run the signal wrt semaphore.

4. a. no because it does not protect the swap from being interrupted. If thread_1 was executing the swap and was interrupted by a time slicing algorithm by thread_2, then thread_2 starts executing swap and changes the value of local it would cause problems when thread_1 began executing again.
b. it is reentrant since all values would be maintained when the Process is interrupted.

5. I would create a named pipe that is shared between P1 and P2 then create a semaphore called preventP2 and a counter variable called P1work. As P1 increments the P1work counter it would attain the PreventP2 semaphore and put P2 to sleep. The when P1 is done working and ready to wake P2 I would decrement P1work count and signal(PreventP2) waking P2. This would require setting P2 to wait on similar to the readers writers problem. At this point the awakened P2 would recieve the file transmitted through the pipe and do some sort of compression on it (e.g. .zip, .tgz, .tar.gz)

6.
```
 Monitor MathFun {
         condition c1, c2, c3;
         private int V1, V2, V3;

         public function increment() {
                 c1.wait();
                 v1++;
                 c1.signal;
         }

         public function decrement() {
                 c1. wait()
```

```
                v1--;
                c1.signal();
        }
        public function square() {
                c2. wait();
                v2*=v2
                c2.signal();
        }
        public function squareRoot() {
                c2. wait();
                v2= sqrt(v2);
                c2.signal();
        }

        public function sin() {
                c3. wait();
                v3 =sin(v3);
                c3.signal();
        }

        public function cos() {
                c3. wait();
                v3=cos(v3);
                c3.signal();
        }
}
```