Homework 4

Problem 1:

Show that the following operations of a simplified bank account application are not correct for multiple threads. Use the notation $r_i(x = 10)$ to denote the event of thread i reading variable x and obtaining the value 10 and $w_i(x = 10)$ for the event of thread i writing to variable x the value 10.

```
class Account:
      value: int

def deposit(account: Account, amount: int):
      account.value += amount

def retrieve(account: Account, amount: int):
      account.value -= amount

def transfer(source_account: Account,
             destination_account: Account,
             amount: int):
      source_account.value -= amount
      destination_account.value += amount
```

You will need to show a history of operations that results in bad values.

Problem 2:

Hyman thought he solved mutual exclusion and its problems were not recognized for decades. The algorithm is simple. There is an array of booleans `want` that expresses the desire of a thread to enter the critical section. There is also a variable `turn` that indicates which thread can enter the critical section. The simplified code is

```
want = [False, False]
turn = 0

def hyman():
      id = thread_id()  # am I thread 0 or thread 1
      want[id] = True
      while( turn != id):
            while want[1-id] == True:
                  pass #waiting for the other
            turn = id
      #Critical Section
      want[id] = False
```

I left out the possibility to re-enter the critical section several times.

(a) Write three different histories of Hyman's using the notation of Problem 1.
(b) Give on history that shows that both threads can end up in the same section.

Hint for (b): Start out with the following history:

> w_1(want$[1]$ = True)
> r_1(turn $= 0$)
> r_1(want$[0]$ = False)
> w_0(want$[0]$ = True)
> r_0(turn $= 0$)
> Thread 0 enters the critical section and sleeps
> Thread 1 wakes up and now does what?