CS 4410 Operating Systems
Prelim I, Fall 2013
Profs. Sirer and Van Renesse

Name:_________________________________ NETID:________________

- **This is a <u>closed book</u> examination. It is 10 pages long. You have 120 minutes.**
- **No electronic devices of any kind are allowed.**
- **If you are taking this exam during the makeup period, you may not leave the exam room prior to the end of the exam, and you may not take an exam booklet with you.**
- **You may use Python, C, or low-level pseudo-code in answer to any coding question. Descriptions written mostly in English will get no credit as responses to coding questions. In addition to correctness, elegance and clarity are important criteria for coding questions.**
- **Assume Mesa semantics for all monitors. Show your work for partial credit. Brevity is key.**
- **Follow the Twelve Commandments**

[20] **1.**

a. Explain how a hardware interrupt is handled from start to finish (approximately 10 sentences)

b. In modern operating systems, an interrupt is often handled in two stages: a first stage during which interrupts are disabled, and a second that is scheduled at a later time (similar to a thread), during which interrupts are enabled. Explain why this may be. (Approximately 5 sentences.)

c. In a microkernel operating system, device drivers may run as a normal process in user space. Discuss how such a driver may access the device and handle device interrupts. (Approximately 5 sentences.)

d. Discuss advantages and disadvantages of running device drivers in user space. (Approximately four advantages and disadvantages.)

[20] **2.** Use monitors and Mesa-style condition variables to implement semaphores.

```
Class Semaphore:
    def __init__(self, count):







   def P(self):







   def V(self):
```

[20] **3**.
You have been hired by Fauxbook Inc to optimize their social networking site. Ever since the site went viral, users have been complaining that adding friends is a very slow operation.

Below, you will find the relevant code snippets for Fauxbook as it is today. The code for the class FriendList is not given to you, but you are told that it exports an "add()" method, and that at most one thread should be calling add() on a given instance of a FriendList at any time. If a user is already on a FriendList, a subsequent addition is a no-op. There is no way to remove a friend. You are similarly not given the code for GetUniqueId(), but you are told that it is implemented by a self-contained monitor that returns unique integers.

As you can see, the code currently relies on a single global lock, which means that at most one user can be adding a friend at any one time. Your task is to change to a more concurrent design where users can add friends concurrently.

Your solution should have the following properties:

- Style: use monitors and Mesa-style condition variables,
- Safety: ensure that all friend relationships are symmetric, i.e. if user A appears on B's friends list (as observed by a thread calling the show_friends() method), then user B also appears on A's friends list (observed similarly).
- Liveness: make progress whenever it is possible to do so, and provide maximum concurrency,

Current Fauxbook Code:

```
dabiglock=Lock()

class User:

    def __init__(self):

        self.userid = GetUniqueId()
        self.listoffriends = FriendList()




   def show_friends(self):
        global dabiglock

        with dabiglock:
             return copy(self.listoffriends)



   def add_friend(self, friend):
        global dabiglock

        with dabiglock:
             self.listoffriends.add(friend)
             friend.listoffriends.add(self)
```

Your Improved Fauxbook Code:

```
class User:

    def __init__(self):

        self.userid = GetUniqueId()
        self.listoffriends = FriendList()




   def show_friends(self):




   def add_friend(self, friend):
```

[20] **4.** Following your success at Fauxbook, you are brought in to write a simulator for an online dating website.

In this simulator, there are four kinds of people, men seeking men, men seeking women, women seeking men and women seeking women, each corresponding to a thread. These threads enter a DatingArea, where they need to be matched with another thread of the gender that they are seeking.

Your task is to use monitors and Mesa-style condition variables to implement the DatingArea. Each thread (named he_he, he_she, she_he, she_she, respectively) calls a specific entry routine. You need to ensure that this routine does not return unless that thread has been matched with a unique thread of the right gender. Your solution must also make progress whenever it is possible to do so.

```
class DatingArea:
    def __init__(self):









    # Invoked by a he_he thread that is looking to match up with another he_he thread
    def he_he_meet(self):
```

```
# Invoked by a he_she thread that is looking to match up with a she_he thread
def he_she_meet(self):
```

```
# Invoked by a she_he thread that is looking to match up with a he_she thread
def she_he_meet(self):
```

```
# Invoked by a she_she thread that is looking to match up with another she_she thread
def she_she_meet(self):
```

Hint #1: if your solution is symmetric for he's and she's, you may provide code for just the he_he_enter() and he_she_enter() routines (or vice versa).
Hint #2: there exists a solution that is symmetric for he's and she's.

[20] **5.** Suppose you have the following tasks for execution on a single core processor. The scheduler knows the duration of each task.

Task Name	Duration	Arrival Time
A	40	0
B	20	10
C	5	20
D	25	30

a. i. Without preemption, what are the completion times and response times for the four tasks if using FIFO and Shortest Job First? Recall that completion time is defined as the time from a task's arrival to the time the task is finished and the response time is defined as the time from a task's arrival to the time that task is first scheduled.

Task	FIFO Completion Time	FIFO Response Time	SJF Completion Time	SJF Response Time
A				
B				
C				
D				

a.ii. What are the average response times for FIFO and SJF, respectively, for these tasks?

a. iii. Define the "burden" of a process as the difference between response time and task length, i.e. (response time - task length). Which processes were most disadvantaged for FIFO and SJF, respectively?

b. i. With preemption every 5 seconds, what are the completion times and response times for the four tasks when using Shortest Remaining Time First?

Task	SRTF Completion Time	SRTF Response Time
A		
B		
C		
D		

b.ii. What is the average response time using SRTF for these tasks?

b. iii. Which processes were most disadvantaged by SRTF?