CS 4410 Operating Systems Final Fall 2010, Prof. Sirer

Name:_____ NETID:_____

							3	

- **This is a closed book examination. It is 10 pages long. You have 150 minutes.**
- **No electronic devices may be used during the exam.**
- **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. Show your work for partial credit. Brevity is key.**

[10 points] 1. **Synchronization**

Some processors do not support the test-and-set instruction, but instead provide ATOMIC-INC and ATOMIC-DEC instructions that atomically increment and decrement the value at a memory location, respectively, and return the value at that location prior to the increment or decrement. Using the following primitives built around these instructions:

```
int atomic_inc(int *location); // incr and return old value
int atomic_dec(int *location); // decr and return old value
```

as well as the following primitives to manipulate thread state and queues:

```
TCB *thread_self(); // get the TCB for current thread
void thread_yield();  // yield the CPU

// block the calling thread
void thread_block();
// block the caller and increment the value at location atomically
void thread_block_and_incr(int *location);
// block the caller and decrement the value at location atomically
void thread_block_and_decr(int *location);
// unblock the named thread
void thread_unblock(TCB *thread);

// enqueue and dequeue
void queue_enqueue(Queue *q, void *element);
void *queue_dequeue(Queue *q);
```

implement semaphores, making sure to provide a type definition, and the implementation of P() and V() operations. (Hint: you may want to start with an implementation based on test-and-set, and modify it for atomic increment and decrement).

[15 points] 2. **Synchronization**
Your task is to write a concurrent simulator for a very small car manufacturing plant. A car is made in four different stages. First, a chassis is welded together. Then, the engine is placed on the chassis. The tires are attached next. And finally, the body is bolted on.

These four operations (namely, chassis, engine, tires and body) are carried out by four workers, named Alice, Bob, Charlie and Debbie, each of whom are trained to perform a subset of the tasks. Alice can build a chassis; Bob can do the engine or the body; Charlie can attach the tires; and Debbie can bolt on the body.

Your job is to write part of the code for modeling the factory floor. We have provided the code for the four workers, which you should not (need to) modify. For simplicity, assume that cars spontaneously combust and disappear the moment they are finished (they are Ford Pintos), obviating the need to model what happens to completed cars.

Use monitors and condition variables in your solution, keeping in mind that the underlying implementation uses Mesa-style semantics. Your solution should be correct, make progress whenever possible to do so, and exhibit efficiency, concurrency, simplicity and elegance. Extracting the correctness criteria is part of the problem. Your solution must avoid busy-waiting for Bob, Charlie and Debbie (Alice does not wait, but she sleeps).

```
from threading import Thread
from threading import Lock, Condition
import time

CHASSIS, ENGINE, TIRES, BODY = 0, 1, 2, 3

class FactoryFloor():
    def __init__(self):
        self.lock = Lock()
        self.floor = []
```

```
      # find a car in a given state and take it off the floor
    def find_car(self, state):
        for i in range(0, len(self.floor)):
            if self.floor[i].state == state:
                item = self.floor[i]
                del self.floor[i]
                return item
        print "FAIL", state
```

```
def alice_done(self, car):




def bob_ready(self):




def bob_done(self, car):




def charlie_ready(self):




def charlie_done(self, car):




def debbie_ready(self):
```

```
ff = FactoryFloor()

class Car:
    def __init__(self):
        self.state = CHASSIS

class Alice(Thread):
    def run(self):
        while True:
            # create a car chassis and place it on the
            # factory floor
            car = Car() # builds a car
            ff.alice_done(car)
            time.sleep(10) # Alice takes a break after each car

class Bob(Thread):
    def run(self):
        while True:
            car = ff.bob_ready()
              # either put the engine on or bolt on the body
            car.state += 1
            ff.bob_done(car)

class Charlie(Thread):
    def run(self):
        while True:
            car = ff.charlie_ready()
             # put the tires on
            car.state += 1
            ff.charlie_done(car)

class Debbie(Thread):
    def run(self):
        while True:
            car = ff.debbie_ready()
            # put the body on
           car.state += 1
            # car is done

alice = Alice()
bob = Bob()
charlie = Charlie()
debbie = Debbie()
alice.start()
bob.start()
charlie.start()
debbie.start()
```

[10 points] 3. Page Replacement

Consider a computer with an address space of size 2^{19} bytes. Each page on this computer is 2^{16} bytes big. This computer has only 2^{18} bytes of physical memory. Consider the following sequence of accesses, where each value in the sequence below refers to a virtual page number:

1,2,3,4,5,1,2,3,4,5,1,2,3,4,5

a. Assuming that memory starts empty, how many page faults will occur and what will be the final contents of memory under the FIFO page replacement policy?

b. Assuming that memory starts empty, how many page faults will occur and what will be the final contents of memory under the LRU page replacement policy?

c. Describe a history-based algorithm, that is, an algorithm that is oblivious of future memory accesses, that outperforms LRU for this particular sequence. (*Hint*: OPT is not a history-based algorithm). How many page faults does it incur?

[30 points] **4. Networking**

a. Some researchers have proposed TCP variants that respond to congestion events by reducing their window size by a small constant amount. Describe, in at most three sentences, how such protocols would interact with existing TCP implementations.

b. Most video distribution sites (like YouTube) and real-time radio stations transport their content using TCP. Is TCP a suitable transport protocol for timely multimedia content that can tolerate loss? Why or why not?

c. Automated teller machines (ATMs) are essentially small computers that act as RPC clients to a bank's servers. They perform ATM transactions using the reliable TCP protocol and implement *at-most-once* semantics. The pseudo-code for one such machine from the Cheapo-ATMs-Are-Us Corporation is shown below. The code for "dispensecash" is not shown, but it instructs the cash counter to dispense the desired amount, and can be assumed to always work correctly, without raising any exceptions. Find and describe the error (Do not attempt to fix it. Hint: the fix is non-trivial).

```
bank = RPCConnection(bank_ipaddress, bank_port)
class ATMMachine(Thread):
      def withdraw(self, cardnumber, amount):
         amount += 2.50    # add our exorbitant fee
         succeeded = False
         with self.lock:
            while not succeeded:
                  try:
                        # the following call is an RPC call to the
                        # bank server to adjust the card holder's
                        # account balance downward
                        bank.debit(cardnumber, amount)
                        # now dispense the cash
                        self.dispensecash(amount)
                        succeeded = True
                  except RPCFailed:
                        print "Cannot communicate with bank server"
```

d. Describe the operation of, and the amount of state kept in each node for, a link-state routing protocol.

e. Describe why there is a minimum packet size limitation on Ethernet networks.

f. What is the purpose of the DHCP protocol?

[15 points] **5. Disks and RAID**

a. What is the advantage of RAID-1 (mirroring) compared to RAID-5 (block-striped parity)?

b. What is the disadvantage of RAID-1 (mirroring) compared to RAID-5 (block-striped parity)?

c. Some disk controllers enable the operating system to issue *dependent operations*, that is, the OS can issue a single operation "B after A", in which case the disk controller guarantees that B will not be performed until and unless A has completed successfully. Describe how a filesystem might use this feature. Be specific, and name the types of blocks and operations for which this feature can be useful. Describe why this feature may be desirable over the simpler method of not issuing an operation (say, B) until a previous one has completed (say, A).

[10 points] 6**. Filesystems**
a. [7 points] Consider a filesystem containing the following directories and files:

/ - root directory
/stuff - a file containing 1000 bytes
/email - a directory
/email/msg – a file containing 3000 bytes

Assuming a block is 1024 bytes, a directory entry is 100 bytes, an inode contains two direct blocks, an indirect block can point to 10 data blocks, draw the contents of all the blocks comprising this file system. Be sure to label each block with its type.

b. [3 points] Briefly, in no more than three sentences, describe the advantages of a log-structured filesystem over a traditional Unix-style filesystem.

[10 points] 7**. Security**
a. What is the difference between authentication and authorization? Can the two terms be used interchangeably?

b. Suppose you're in charge of guarding the secret files containing the communication between US diplomats and the state department (like the ones that found their way onto WikiLeaks). Assume that all these files are kept on a single filesystem, where all users have an account and need to have access to different subsets of the files individually tailored to what they need to know (i.e. there are no "group permissions" because there are no meaningful groups; representing a full, general-purpose access control matrix is called for).

If it were the case that the classification level for the files was changing frequently, would you prefer to use capabilities or access-control lists? Why or why not?

[3 extra points] **8. Futuristic Security**
In futuristic movies involving computers:

Anyone can type "DOWNLOAD ALL SECRETS" to download all the secrets.
"Permission Denied" messages have a convenient "OVERRIDE" button.
It is trivial to load a virus onto alien computers.

because:

a. We will inevitably forget everything we know about computer security in the near future
b. The Facebook generation will have no concept of private information, and won't care
c. Aliens are using an old version of IE6 to surf the Internet
d. All of the above