University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Fall 2011 Anthony D. Joseph and Ion Stoica

# Midterm Exam *Solutions*

October 13, 2011
CS162 Operating Systems

| | |
|---|---|
| **Your Name:** | |
| **SID AND 162 Login:** | |
| **TA Name:** | |
| **Discussion Section Time:** | |

General Information:
This is a **closed book and one 2-sided note** examination. You have 80 minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points for that question. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. ***Make your answers as concise as possible.*** If there is something in a question that you believe is open to interpretation, then please ask us about it!

**Good Luck!!**

| QUESTION | POINTS ASSIGNED | POINTS OBTAINED |
|---|---|---|
| 1 | 12 | |
| 2 | 42 | |
| 3 | 18 | |
| 4 | 28 | |
| TOTAL | 100 | |

1. (12 points total) Short answer questions:

a. (4 points) List the four requirements for deadlock:

*Mutual exclusion, non-preemption, hold and wait, circular wait.*

*We gave one point for each requirement for deadlock.*

b. (3 points) Operating System terms.

i) (1 point) List a synchronization mechanism that is an abstraction of a counting number (non-negative integer).

*A (counting) semaphore.*

*We gave 1 point for correctly described term; in very few cases we gave partial credit.*

ii) (1 point) What is the portion of a process' code that manipulates shared variables?

*A critical section.*

*We gave 1 point for correctly described term; in very few cases we gave partial credit.*

iii) (1 point) What is the OS data structure that represents a running process?

*The Process Control Block.*

*We gave 1 point for correctly described term; in very few cases we gave partial credit.*

c. (5 points) What are the differences between a Hoare monitor and a Mesa monitor? Your answer should be brief (no more than 3 sentences).

*Hoare monitors suspend the signaling thread, and transfer the lock and control to the thread being woken up. Mesa monitors simply schedule the thread being woken up and release the lock. With Mesa monitors, the thread being woken up must contend for the lock and the resource(s) being protected by the critical section..*

*For Hoare Monitors we gave 1 point for transfer lock, 1 point for transfer cpu, and 1 point to thread being woken.*

*For Mesa Monitors, we gave 1 point for putting the thread into ready queue, and 1 point for rechecking the condition and content for resources (1pt)*
*We subtracted 1pt for mentioning that the lock and cpu are not transferred for Mesa monitors.*

2. (42 points total) CPU Scheduling.

a. (4 points) Provide a brief definition of what are preemptive and non-preemptive scheduling policies. Your answer should not exceed four sentences.

*With the non-preemptive scheduling policy, scheduling only occurs when a process enters the wait state or terminates. With the preemptive scheduling policy, scheduling also occurs when a process switches from running to ready due to an interrupt and from waiting to ready (i.e., the event a thread has been waiting occurs – I/O completion).*

*We gave 2 points for defining preemptive scheduling (i.e., "scheduler can take away CPU from the job before the CPU burst is over), and 2 points for defining the nonpreemptive scheduling (i.e., "jobs run until they explicitly yield the resource").*

*We took away 1 point for each obviously incorrect statement, e.g. "preemptive scheduling tries to minimize job completion time".*

b. (3 points) Why is SJF or SRTF scheduling difficult to implement in a real OS? Give an alternative approach *with the same goals* that can be implemented easily in a real OS

*These algorithms require knowledge of the future. In real systems, we can guess but cannot know what a program will do next. An approximation is to look at each process' history as a likely indicator of its future behavior (i.e., CPU burst length)*

*We gave points for saying "it's hard to predict the future", and 1 point for giving Multilevel Feedback Queue as an example. We didn't accept lottery scheduling.*

c. (35 points) Here is a table of processes and their associated arrival and running times.

| **Process ID** | **Arrival Time** | **Expected CPU Running Time** |
|---|---|---|
| Process 1 | 0 | 4 |
| Process 2 | 2 | 5 |
| Process 3 | 3 | 3 |
| Process 4 | 8 | 4 |

i) (15 points) Show the scheduling order for these processes under First-In-First-Out (FIFO), Shortest-Job First (SJF), and Round-Robin (RR) with a quantum = 1 time unit. *Assume that the context switch overhead is 0 and new processes are added to the* ***head*** *of the queue except for FIFO.*

| **Time** | **FIFO** | **SJF** | **RR** |
|---|---|---|---|
| **0** | *1* | *1* | *1* |
| **1** | *1* | *1* | *1* |
| **2** | *1* | *1* | *2* |
| **3** | *1* | *1* | *3* |
| **4** | *2* | *3* | *1* |
| **5** | *2* | *3* | *2* |
| **6** | *2* | *3* | *3* |
| **7** | *2* | *2* | *1* |
| **8** | *2* | *2* | *4* |
| **9** | *3* | *2* | *2* |
| **10** | *3* | *2* | *3* |
| **11** | *3* | *2* | *4* |
| **12** | *4* | *4* | *2* |
| **13** | *4* | *4* | *4* |
| **14** | *4* | *4* | *2* |
| **15** | *4* | *4* | *4* |

ii) (18 points) For each process in each schedule above, indicate the queue wait time and turnaround time (TRT).

The queue wait time is the *total* time a thread spends in the wait queue.
The turnaround time is defined as the time a process takes to complete after it arrives.

| Scheduler | Process 1 | Process 2 | Process 3 | Process 4 |
|---|---|---|---|---|
| FIFO queue wait | *0* | *2* | *6* | *4* |
| FIFO TRT | *4* | *7* | *9* | *8* |
| SJF queue wait | *0* | *5* | *1* | *4* |
| SJF TRT | *4* | *10* | *4* | *8* |
| RR queue wait | *4* | *8* | *5* | *4* |
| RR TRT | *8* | *13* | *8* | *8* |

*Part a) 5 points per column, part b) 3 points per row. We deducted one point per error up to the maximum score for the column/row. We deducted one point for a wrong overall arrival time.*

iii) (2 points) Which of these three scheduling disciplines has the lowest average TRT?

*SJF. You don't have to do any calculations to know that.*

*No Credit* – **Problem X** (000000000000 points)

## The 2010 Stella Awards

The annual 'Stella Awards' are named after 81-year-old Stella Liebeck who spilled hot coffee on herself and successfully sued the McDonald's in New Mexico, where she purchased coffee. You remember, she took the lid off the coffee and put it between her knees while she was driving. Who would ever think one could get burned doing that, right?

***SEVENTH PLACE***
Kathleen Robertson of Austin, Texas was awarded $80,000 by a jury of her peers after breaking her ankle tripping over a toddler who was running inside a furniture store. The store owners were understandably surprised by the verdict, considering the running toddler was her own son.

***SIXTH PLACE ***
Carl Truman, 19, of Los Angeles, California won $74,000 plus medical expenses when his neighbor ran over his hand with a Honda Accord. Truman apparently didn't notice there was someone at the wheel of the car when he was trying to steal his neighbor's hubcaps.

*** FIFTH PLACE ***
Terrence Dickson, of Bristol, Pennsylvania, who was leaving a house he had just burglarized by way of the garage. Unfortunately for Dickson, the automatic garage door opener malfunctioned and he could not get the garage door to open. Worse, he couldn't re-enter the house because the door connecting the garage to the house locked when Dickson pulled it shut. Forced to sit for eight, count 'em, EIGHT days and survive on a case of Pepsi and a large bag of dry dog food, he sued the homeowner's insurance company claiming undue mental Anguish. Amazingly, the jury said the insurance company must pay Dickson $500,000 for his anguish.

***FOURTH PLACE***
Jerry Williams, of Little Rock, Arkansas, garnered 4th Place in the Stella's when he was awarded $14,500 plus medical expenses after being bitten on the butt by his next door neighbor's beagle - even though the beagle was on a chain in its owner's fenced yard. Williams did not get as much as he asked for because the jury believed the beagle might have been provoked at the time of the butt bite because Williams had climbed over the fence into the yard and repeatedly shot the dog with a pellet gun.

***THIRD PLACE ***
Amber Carson of Lancaster, Pennsylvania because a jury ordered a Philadelphia restaurant to pay her $113,500 after she slipped on a spilled soft drink and broke her tailbone. The reason the soft drink was on the floor: Ms.Carson had thrown it at her boyfriend 30 seconds earlier during an argument.

***SECOND PLACE***
Kara Walton, of Claymont, Delaware sued the owner of a night club in a nearby city because she fell from the bathroom window to the floor, knocking out her two front teeth. Even though Ms. Walton was trying to sneak through the ladies room window to avoid paying the $3.50 cover charge, the jury said the night club had to pay her $12,000....oh, yeah, plus dental expenses.

***FIRST PLACE ***
This year's runaway First Place Stella Award winner was: Mrs. Merv Grazinski, of Oklahoma City, Oklahoma, who purchased new 32-foot Winnebago motor home. On her first trip home, from an OU football game, having driven on to the freeway, she set the cruise control at 70 mph and calmly left the driver's seat to go to the back of the Winnebago to make herself a sandwich. Not surprisingly, the motor home left the freeway, crashed and overturned. Also not surprisingly, Mrs. Grazinski sued Winnebago for not putting in the owner's manual that she couldn't actually leave the driver's seat while the cruise control was set.. The Oklahoma jury awarded her, $1,750,000 PLUS a new motor home. Winnebago actually

changed their manuals as a result of this suit, just in case Mrs. Grazinski has any relatives who might also buy a motor home.

3. (18 points) Networking.

a. (5 points) In a network, both the link layer and the transport layer may provide error detection facilities. Give 2 reasons that a transport-level error detection mechanism is needed to ensure end-to-end data integrity.

*The link layer is not guaranteed to provide error detection, and the transport can't be sure which link levels it's running over. Even if all the link levels provide error detection, this does not detect errors in the intermediate systems in the network.*

*We gave 2.5 points per reason Two reasons that were basically the same idea counted as one reason. Additional acceptable answers (+2.5 each):*

- *Data could be corrupted, lost, etc. at higher layers than link layer*
- *Can't check for reordering of packets with some explanation of why this is true (packets can take different links, etc.)*
- *Enforcing at the link layer would impose unnecessary burden on applications that don't need reliability*

*Only endpoint can guarantee that the endpoint received a packet Partially acceptable answers (+1 each, more or less depending on the case):*

- *Explanation of E2E principle without tying it back to the actual issue*
- *Talking about reordering of packets, but inadequate explanation of what could go wrong with that*
- *Mentioning that endpoints would "have to check anyway" but no explanation of why*

*Unacceptable answers (+0 each):*

- *Just stating that packets can be dropped / corrupted / reordered / etc. in general*
- *Explanations of what reliability means*
- *Very general statements about network architecture*
- *Link layer "doesn't have enough information" with no / inadequate additional explanation*
- *Saying that reliability is the transport layer's responsibility, or that it's just the way the network is made*
- *Comparing transport layer vs. application layer (instead of link layer)*
- *Not mentioning anything about the link layer at all - Saying that the transport layer is in charge of routing*
- *Saying that the transport layer can ask for packets to be resent, with no additional explanation*

b. (8 points) Statistical Multiplexing.

i) (5 points) Briefly explain the concept of statistical multiplexing. Your answer should not exceed three sentences.

*As the number of flows increases in a network, the ratio of the aggregate peak bandwidth to the aggregate average bandwidth decreases.*

We gave 2 points for the idea of multiple flows/flows with different peaks or burst times, and 3 points for the idea that as the number of flows increases, the peak to the average ratio decreases.

ii) (3 points) Briefly explain why statistical multiplexing is helpful for network designers. Your answer should not exceed two sentences.
*Multiplexing a large enough number of flows "eliminates" burstiness and enables you to use the average bandwidth to provision capacity, instead of peak bandwidth.*

*We gave one point for the idea that multiplexing many flows "reduces burstiness"/smooths out peaks, and 2 points for the idea that this means that network designers need to provision less bandwidth, i.e., slightly over the average, rather than the peak.*

c. (5 points) Flow control. Assume a sender sending 1000 byte packets to a receiver with a window size of 10 packets. Assume the round-trip time (RTT) is 50ms. What is the maximum throughput (i.e., bits/sec) that the sender can sustain? (Ignore the packet header in your answer.)
*The sender can send up to 10 packets during an RTT. Thus, the maximum throughput (R) is:*

$R = (1000\text{byte}*8\text{bits}*10\text{ packets})/(5*10^{-2}\text{sec}) = 1{,}600{,}000\text{ bps} = 1.6\text{ Mbps}$

*We gave 5 points for correct expression and answer; answers with correct expression and correct unit conversion but with minor arithmetic errors also received full credit. We subtracted 2 points for failing to account for receiver window, 3 points for incorrect RTT, and 1 point each for unit conversion errors (msec to sec, bytes to bits etc).*

4. (28 points total) Concurrency control.
   a. (12 points total) Dining Philosophers.
   The goal of this exercise is to implement a solution to the Dining Philosophers problem – 5 philosophers sitting at a round table. Philosophers repeat (forever) the following three things in order: (1) think, (2) eat, and (3) sleep. Each philosopher has a plate of spaghetti in front of him or her and a chopstick between each plate.

   **Important rule:** A philosopher must have two (2) chopsticks to be able to eat! A philosopher must acquire the chopstick to the left of their plate and the chopstick to the right of their plate; they cannot reach across the table to use other chopsticks. When they are done eating, they put down (and free) the two chopsticks for someone else to use. Multiple philosophers should be able to acquire chopsticks at the same time.
   Below is code that attempts to solve this problem. Each philosopher will call this method with the parameter (0 to 4) indicating which philosopher they are. Unfortunately, this code can get stuck (deadlock).
   *You need to fix this code so it cannot deadlock and so that more than one philosopher can eat at the same time.*

   The constructor `Philosopher()` is called once to create the 5 chopsticks and 5 philosopher threads are created, each one executing the `Dine()` method.

   `Think()`, `Eat()`, and `Sleep()` are methods provided to you (already written). You do not know how long each of these takes to complete.

   *Your solution must avoid deadlock.*

```
Philosopher () {
   for (int i=0; i<=4; i++)
      chopstick[i] = new Semaphore(1);
   for (int j=0; j<=4; j++)
      thread_fork(Dine(j));
}

void Dine(int p) {
   while (1) {
     Think();
     chopstick[p].P();
     chopstick[(p + 1) % 5].P();
     Eat();
     chopstick[(p + 1) % 5].V();
     chopstick[p].V();
     Sleep();
    }
}
```

i) (4 points) Specify the correctness constraints. Be succinct and explicit in your answer.
*A diner waits for two chopsticks (2 points).*
*The chopsticks must be the ones immediately to the diner's right and left. (2 points)*

*We gave one point for stating some other type of related constraint (e.g., you can't steal chopsticks, there's one chopstick per diner, you can't take a chopstick from across the table, etc.) We subtracted one point for saying something that is implementation dependent or otherwise not a correctness constraint.*
*We gave zero points for answers that described only an implementation (e.g., each philosopher must first pick the left chopstick) instead of correctness constraints.*

ii) (8 points) Correctly re-implement the `Dine()` method.
*The key insight is to have every other philosopher get the first chopstick in the opposite order, which breaks the cyclic dependency.*
.

```
void Dine(int p) {
   while (1) {
     Think();
     // By having every other philosopher get the chopsticks
     // in the opposite order, we break the cyclic
     // dependency
     if (p%2 == 0) {
       // Grab left, then grab right
       chopstick[p].P();
       chopstick[(p + 1) % 5].P();
     } else {
       // Grab right, then grab left
       chopstick[(p + 1) % 5].P();
       chopstick[p].P();
     }
     Eat();
     chopstick[(p + 1) % 5].V();
     chopstick[p].V();
Sleep();
   }
}
```

*If the solution made use of locks, we subtracted 4 points for excessive locking, as two philosophers cannot pick chopstick simultaneously.*
*We subtracted six points if your solution could still deadlock and 1 point if your solution was missing think/eat/sleep. We took four points off if your solution broke or violated the correctness constraints*

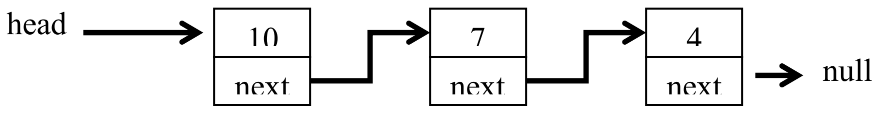
b. (10 points) Synchronization. The following Java code implements three operations to handle a linked list: (a) insertion at the head of the list, (b) deletion from the head of the list, and (c) counting and printing the number of the items in the list.

```
void insertHead(int value) {
   item = new Item(value);
   if (head == null) {
      head = item;
   } else {
      item->next = head;
      head = item;
   }
}

item deleteHead(int value) {
   item result = head;
   if (head != null) {
      head = head->next;
   }
    return result;
}
```

```
int getLen() {
   int len = 0;
   item tmp = head;
   while (tmp) {
      len = len + 1;
      tmp = tmp->next;
}
   printf(len);
   return len;
 }
```

Initially, the list contain the following three elements:



Assume there are three threads, each of them performing the following operations. (Note that the code is not thread safe. Assume that only reads and writes from/to memory are atomic.)

| Thread1 | Thread2 | Thread3 |
|---|---|---|
| insertHead(2) | deleteHead() | getLen() |

i) (6 points) What are the possible contents of the lists after the first two threads finish?

*There are four possible outcomes:*

- *10, 7, 4*
- *7, 4*
- *2, 10, 7, 4*
- *2, 7, 4*

*We subtracted 1 point for every missing outcome, and one point for every wrong one.*

ii) (4 points) What are the possible outputs printed by Thread 3?

*There are three possible output values: 2, 3, 4*

*We subtracted 1 point for every missing outcome, and one point for every wrong one.*

c. (6 points) We discussed the Therac-25 in lecture and in an assigned reading.

i) (3 points) Explain the Therac-25's problems and the underlying causes.

*Software errors caused the deaths and injuries of several patients. There were a series of race conditions on shared variables and poor software design that lead to the machine's malfunction under certain conditions.*

*We gave full points for specifying at least one of the variety of causes listed in the paper.*

ii) (3 points) What would have been a good software development model for the Therac-25's manufacturer to use?

*The Waterfall model. The Therac-25 had clear design requirements and the significant checks in the waterfall model would have been a better alternative to the ad hoc development model that was used.*

*We gave full points for every answer that mentioned "waterfall". Otherwise, we gave 1 point if the answer mentioned testing.*