COM S 352: Introduction to Operating Systems
Midterm Practice Exam
Fall 2022

Cover Sheet

Student Name: ____

Format:

- Time: 50 mins
- Points: 100
- Question Types: matching, true/false and short answer

Instructions:

- You may use 1 (one) letter sized sheet of paper (front and back), that you have prepared yourself with notes before the exam, as a "cheat sheet" during the exam.
- You may not consult classmates, electronic devices or resources other than the cheat sheet during the exam.
- Questions of clarification should be asked directly to an instructor or TA.

Question	**Points**
1	/24
2	/30
3	/10
4	/10
5	/6
6	/6
7	/10
8	/4
Total	/100

1. (24 pts, 3 pts each) For each description on the left, select the best matching term on the right, each term is used only once but some will not be used.

__i__ makes a copy of the current process

__a__ a process that frequently exceeds its time slice

__h__ mechanism that can provide both locking and signaling

__d__ created as a result of calling pthread_create()

__e__ system calls are implemented using traps because they require this

__g__ a queue for inter-process communication

__l__ reason caching can improve performance

__b__ machine instruction that can be used to implement mutex locks

a. CPU bound
b. TestAndSet
c. Heap
d. TCB
e. kernel mode
f. page out
g. pipe
h. semaphore
i. fork()
j. swap
k. MMU
l. locality

2. (26 pts, 2 pts each) Which of the following statements are true? Write T or F for true or false.

__F__ In RR scheduling, best performance is when the time quantum is small with respect to context-switch time.

__F__ In the Linux Completely Fair Scheduler processes with low nice values always run before those with high nice values.

__T__ LRU never experiences Belady's Anomaly.

__T__ The purpose of priority boost is to prevent starvation.

__F__ Thrashing is a result of insufficient multiprogramming.

__T__ A segment can be shared by multiple processes.

__F__ If a page's contents in memory are different than they are on disk, the valid bit must be set to invalid.

__F__ An illegal memory access results in a page fault.

__F__ A translation lookaside buffer (TLB) is used to search for free space in physical memory.

__T__ Paging has the problem of internal fragmentation.

__T__ Coalescing nodes in a free space list is done to make it easier to search for contiguous regions of free memory.

__F__ Multi-level pages tables are used to increase page table lookup speed.

__T__ The bounded-buffer problem can be solved using only semaphores to control concurrency.

__T__ A binary semaphore is equivalent to a mutex lock.

~~__T__ A thread requesting a resource never causes a deadlock as long as that thread does not currently have any other resources assigned to it.~~

3. (10 pts) Consider the following set of jobs, with arrival times and the length of CPU bursts given in milliseconds.

	Arrival Time	CPU Burst
A	0	10
B	1	12
C	4	10
D	5	2

Create Gantt charts and compute the average response time for each of the following scheduling algorithms. Show your calculations.

a) SJF (without preemption)

Gantt chart:

```
| A | D | C | B |
0   10  12  22  34
```

Average response time is __8.5__ (show calculation below)

(0 + (10 – 5) + (12 – 4) + (22 – 1)) / 4 = 8.5

b) STCF

Gantt chart:

```
| A | D | A | C | B |
0   5   7   12  22  34
```

Average response time is __7.25__ (show calculation below)

(0 + (5 – 5) + (12 – 4) + (22 – 1)) / 4 = 7.25

4. (10 pts) Given the reference string of page accesses: 4 3 2 1 4 3 5 1 and a system with 3 page frames, how many page faults result when using the following page replacement policies? Show your work.

a) FIFO

Number of page faults is __8__ (show work below)

Page ref	4	3	2	1	4	3	5	1
H/M	M	M	M	M	M	M	M	M
cache			4	3	2	1	4	3
cache		4	3	2	1	4	3	5
cache	4	3	2	1	4	3	5	1

b) OPT

Number of page faults is __5__ (show work below)

Page ref	4	3	2	1	4	3	5	1
H/M	M	M	M	M	H	H	M	H
cache			4	4	4	4	5	5
cache		4	3	3	3	3	3	3
cache	4	3	2	1	1	1	1	1

5. (6 pts) The code for the xv6 kernel function sched() is provided below.

```
void
sched(void)
{
  // ...
  struct proc *p = myproc();
  // ...
  swtch(&p->context, &mycpu()->context);
  // ...
}
```

Why does the function call swtch()? What does swtch() do? Answer in 2 to 3 sentences.

sched() calls swtch() to perform a context switch from the context of the kernel code that interrupted the currently executing process into the context of the scheduler. The purpose is for the scheduler to possibly context switch to a different process. swtch() performs the actions of a context switch, saving the current CPU registers to the process' context and loading CPU registers from the saved context.

6. (6 pts) Given a virtual address space of 1,024 pages and a 4,096 byte page size, how many bits are required to store a virtual address. Show calculations.

1,024 = 2^10, so 10 bits to store the page number
4,096 = 2^12, so 12 bits to store the offset

10 + 12 = 22

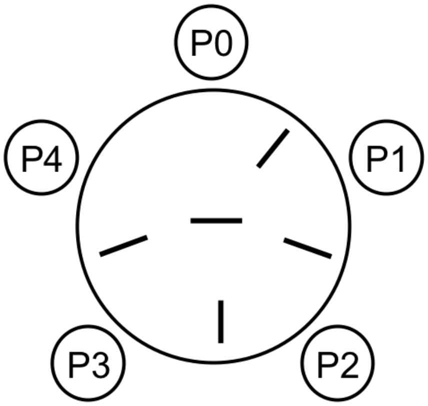
7. (10 pts) Consider two threads ping_thread and pong_thread that output the words “ping” and “pong” respectively. The required output of the program is as follows (it is required that ping always goes first):

ping
pong
ping
pong
…

Provide a solution for ping_thread and pong_thread **using only semaphores** for concurrency control. Declare and initialize all variables. Exact pthread syntax is not required.

```
#include <pthread.h>

// declare any required variables here
sem_t ping;
sem_t pong;


void init() { // perform initialization here

    // there are alternatives, for example, start ping at 1 and have
    // ping_thread wait before printing
    sem_init(&ping, share, 0);
    sem_init(&pong, share, 0);

}


void* ping_thread(void *arg) {
    while (1) { // complete the ping thread

        printf("ping\n");
        sem_post(&pong);
        sem_wait(&ping);

    }
}

void* pong_thread(void *arg) {
    while (1) { // complete the pong thread


        sem_wait(pong);
        printf("pong\n");
        sem_post(&ping);


    }
}
```

8. (4 pts) Consider a modified version of the dining philosophers problem, where one of the chopsticks is placed at the center of the table, available for any of the philosophers to pick up with either hand.



Can deadlock be prevented if no philosopher is allowed to pick up the center chopstick as their first chopstick? Explain you answer.

[NOTE: DINING PHILOSOPHERS PROBLEM WILL NOT BE ON MIDTERM EXAM]

Deadlock is prevented.

The condition that is prevented is circular wait. In the original dinning philosophers problem there is only one possible cycle of philosophers and resources that can form (although it can be directed clockwise or counterclockwise). That cycle is no longer possible for the following reason.

Suppose that 4 of the philosophers each have 1 chopstick. Note that none of them can be holding the center chopstick because it can only be picked up second. The philosopher that isn't holding a chopstick cannot pick up the center chopstick because it can only be picked up second. One of the philosophers will be able to pick up the center chopstick, finish eating and put down two chopsticks. Therefore, there is no set of resource allocation arrows that can be drawn from a philosopher to the center chopstick and back to another philosopher in the resource allocation graph (i.e., no cycle).