Do not turn this page until you have received the signal to start.

(Please fill out the identification section above, write your name on the back of the test, and read the instructions below.)

Good Luck!

This midterm consists of 5 questions on 8 pages (including this one). When you receive the signal to start, please make sure that your copy is complete. Pseudo-code is acceptable where code is required. Answer the questions concisely and legibly. Answers that include both correct and incorrect or irrelevant statements will not receive full marks.

If you use any space for rough work, indicate clearly what you want marked.

TOTAL: _____/31

# 1: _____/10
# 2: _____/ 6
# 3: _____/ 3
# 4: _____/ 5
# 5: _____/ 7
Question 1. Multiple Choice - Mark the best choice for each part [10 marks]

Part (a) [1 mark] When does preemption take place?

☐ When a quantum expires
☐ When a process issues an I/O request
☐ When a process exits
☐ All of the above

Part (b) [1 mark] In contrast to a non-preemptive schedule, a preemptive scheduler supports the following state transition:

☐ Ready → running
☐ Running → ready
☐ Ready → blocked
☐ Blocked → running

Part (c) [1 mark] To implement a user-level threads package, it helps if the operating system provides:

☐ Kernel threads
☐ Non-blocking system calls
☐ An execve mechanism
☐ Direct Memory access

Part (d) [1 mark] Switching between user level threads of the same process is often more efficient than switching between kernel threads because:

☐ User level threads require tracking less state
☐ User level threads share the same memory address space
☐ Mode switching is not necessary
☐ Execution stays within the same process with user level threads

Part (e) [1 mark] Starvation is the case when a thread:

☐ Loops continuously until it runs out of memory
☐ Is never scheduled to run
☐ Can never acquire a lock on a critical section
☐ Cannot create a child process or thread
Part (f)  [1 mark] A thread that is blocked on a semaphore is awakened when another thread

☐ Tries to decrement a semaphore’s value below 0.

☐ Tries to increment the semaphore.

☐ Causes the semaphore’s value to reach a specific number.

☐ Tries to block on the same semaphore.

Part (g)  [1 mark] A quantum is:

☐ The absolute minimum time that a process can run.

☐ The amount of time that a process runs before it blocks on I/O.

☐ The fraction of a time slice during with the process is running.

☐ The maximum time that a process can run before being preempted.

Part (h)  [1 mark] Process aging is:

☐ Computing the next CPU burst time via a weighted exponential average of previous bursts.

☐ The measurement of elapsed CPU time during a process’ execution.

☐ Boosting a process’ priority temporarily to get it scheduled to run.

☐ Giving a process a longer quantum as it gets older.

Part (i)  [1 mark] Which scheduler gives each process an equal share of the CPU?

☐ Round robin

☐ Shortest remaining time first

☐ Priority

☐ Multilevel feedback queues

Part (j)  [1 mark] A multi-level feedback queue generally assigns a long quantum to

☐ High priority processes

☐ Low priority processes

☐ New processes

☐ Old processes
Question 2. For each statement, circle the correct option. [6 MARKS]

TRUE  FALSE  System calls, interrupts and exceptions are all ways to transfer control between the process and the operating system.

TRUE  FALSE  Switching among threads in the same process is more efficient than switching among processes.

TRUE  FALSE  Multilevel queues allow multiple processes to share the same priority level.

TRUE  FALSE  Memory allocated by mymalloc suffers from internal fragmentation.

Consider the following pseudo-code:

1. pthread_cond_t CV;
2. pthread_mutex_t mutex;
3. // Assume CV and mutex are correctly initialized
4. pthread_mutex_lock(mutex);
5. while(count > N) {
6.     pthread_cond_wait(&CV, &mutex);
7. }

TRUE  FALSE  The program assumes Hoare semantics.

TRUE  FALSE  At line 7, this thread holds the lock mutex.
Question 3. [3 marks]
Is the following a valid implementation of the condition variable wait and signal functions? Explain your answer.

```c
void cv_wait(sem_t *s, pthread_mutex_t *l) {
    release(l);
    sem_wait(s);
    acquire(l);
}

void cv_signal(sem_t *s) {
    sem_post(s);
}
```

Question 4. [5 marks]
Given the following four scheduling algorithms: FIFO, SJF (Shortest Job First) (non-preemptive), STCF (Shortest Time to Completion first)(preemptive), Round Robin, assume that jobs arrive over time. Fill in the blanks with the one or more algorithms that satisfy the statements. (It maybe possible that several algorithms are equivalent in some cases.)

Lowest average turnaround time: _________________________

Lowest average response time: _________________________

Lowest overhead cost: _________________________

Highest overhead cost: _________________________

Prevents starvation: _________________________
Question 5.  [7 marks]

Consider the following implementation of mymalloc and myfree. A separate region of N pages of memory is reserved for each thread, and each region has its own freelist. A thread calling mymalloc always searches its own freelist when looking for a chunk of memory to reserve.

Part (a)  [2 marks]

A design issue is how to increase the size of a per-thread region if N pages are used up. Briefly outline a reasonable way to handle this case. (mymalloc returning NULL is not a valid answer unless there really is no more virtual memory that can be reserved.)

Part (b)  [1 mark]

Threads can still share heap memory in this model. Will one thread ever need to access another thread’s freelist? If no, explain your answer. If yes, then give an example of an operation where this might occur.

Part (c)  [2 marks]

Are locks required for the per-thread freelists? Explain your answer.

Part (d)  [1 mark]

Would you expect this algorithm to run faster than the basic algorithm with a single freelist? Explain your answer.

Part (e)  [1 mark]

Compare the per-thread freelist to the single freelist in terms of space requirements.
[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]
Print your name in this box.