CMPUT 379 – Operating System Concepts
Practice Midterm Exam - Fall 2019
Department of Computing Science, University of Alberta
Instructor: Omid Ardakanian

Full Name: ______________________________

Student ID Number: ______________________________

Instructions

- Print your full name and ID clearly above.
- You have 50 minutes to complete the exam.
- There should be 5 questions and 6 pages in this exam booklet. You are responsible for checking that your exam booklet is complete.
- It is a closed-book exam. You are not permitted to bring printed or handwritten notes. Electronic devices such as laptops, calculators, phones, etc. are strictly prohibited.
- Place most answers in the spaces provided on the question pages. Keep your answers brief. Think about each question a bit before answering.
- This exam is worth 100 points and counts 20% toward your final grade in this course. The weight of each question is indicated in square brackets by the question number.

Question	1	2	3	4	5	Total
Out of	20	20	10	20	30	100

Question 1 [20 points]: Operating System Concepts
Choose either True or False for the questions below. You do not need to provide justifications.

(1) The state of a process can change as a result of its actions, Operating System actions, or external events.

☐ True ☐ False

(2) The nicer a process is the more time slots are allocated to it.

☐ True ☐ False

(3) The heap and the stack are shared between threads of a process.

☐ True ☐ False

(4) A synchronous signal is delivered to only one thread of a process.

☐ True ☐ False

(5) Named pipes can be accessed without requiring a parent-child relationship.

☐ True ☐ False

(6) Increasing the time quantum of Round Robin increases the average turnaround time.

☐ True ☐ False

(7) Admission control is necessary in hard real-time systems.

☐ True ☐ False

(8) CPU-bound processes spend more time doing computation than I/O.

☐ True ☐ False

(9) If tasks are variable in size, FCFS is optimal in terms of the average response time.

☐ True ☐ False

(10) The Banker's algorithm is used for deadline detection and recovery.

☐ True ☐ False

Question 2 [20 points]: Short-Answer Questions

Part 2.1 [5 marks]: What causes a process to transition from the running state to the waiting state?

Part 2.2 [5 marks]: List three attributes of a thread that the operating system must maintain when it is not running.

Part 2.3 [5 marks]: List three reasons for the system to go from user mode to kernel mode.

Part 2.4 [5 marks]: Explain why a call to pthread_mutex_lock typically takes longer to return when there is contention for the mutex.

Question 3 [10 points]: Process Control
How many processes are created when this code runs?

```c
#include <unistd.h>
int main () {
  fork();
  fork();
  fork();
  fork();
}
```

Question 4 [20 points]: CPU Scheduling
A uniprocessor system uses the Round Robin scheduling algorithm with the time quantum of 3 milliseconds. Consider the following set of processes, with the length of the CPU burst (given in milliseconds) and the arrival time (given in milliseconds after time 0).

Process	Burst Length	Arrival Time
A	5	0
B	6	2
C	1	5
D	5	9

Given the above information, fill in the following table which shows what process the Round Robin scheduler runs when. Leave cells empty if no process runs on the CPU in those time slots.

Time slot	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10
Process										
Time slot	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20
Process										
Time slot	20-21	21-22	22-23	23-24	24-25	25-26	26-27	27-28	28-29	29-30
Process										

Question 5 [30 points]: Threads & Concurrency
Consider the following code segment which can be executed by one or more threads:

```
x = x + 1
y = x + y
```

We make the following assumptions:

- Both x and y are initialized to 1 before any thread starts running. They are only updated by the threads executing the above code segment.
- A thread can be preempted at any point during its execution.
- Each of the two instructions above is atomic.

Part 5.1 [10 marks]: Assume two threads run the above code concurrently. Write down all possible values of x and y after both threads finish.

Part 5.2 [10 marks]: Same question, but now assume that three threads run this code segment concurrently.

Part 5.3 [10 marks]: How can we avoid race conditions in the above example?