CMSC 412 Midterm #1 (Spring 2014)

1.) (20 points) Define and explain the following terms:

a) Proportionality Principle

Complexity of task should be proportional to rarity/risk (i.e. format disk should be hard to type).

b) Process Control Block (PCB)

A data structure that stores information about a process such as is pid, open files, memory, program its running.

c) CPU protection

Ensuring the OS kernel regains control of a core back from user processes. Often done via a timer interrupt and protecting the interrupt table from user processes updating them.

d) Critical Section

A region of code requiring mutual exclusion, yet ensuring progress is made and a bound waiting time.

2.) (20 points) Given a system that provides binary semaphores (semaphores whose values is either 0 or 1). Show the code to implement counting semaphores using binary semaphores.

```
Sem mutex = 1
Sem gate = min(1,val)
Val = <initial value>
P:
      P(gate)
      P(mutex)
      Val = val -1
      If (val > 0) v(gate)
      V(mutex)

V:
      P(mutex)
      Val = val + 1
      If (val == 1) V(gate)
      V(mutex)
```

3.) (16 Points) Scheduling

a) Given a round robin scheduler (with a quantum of 1 unit), and the following jobs, indicate when each job completes. When a job arrives at time n, it is placed at the head of the queue of processes and can first run at time n+1.

Job	Arrival Time	Required Time	Completion Time
A	0	6	**11**
B	3	3	**9**
C	6	6	**15**
D	15	2	**17**

b) If a scheduler moves processes to a lower priority queue if they use their full quantum, describe a counter strategy for this policy that a process could use to prevent getting put at lower priority.

Yield the processor just an epsilon before the time quantum is up.

4.) (12 points) Deadlock: Explain the four necessary conditions for deadlock.

Circular Wait, Hold & Wait, No Preemption, Mutual Exclusion.

5.) (12 points) Why do we use an interrupt/trap instruction and a number to call a system call rather than simply making it a procedure call?

A trap allow getting to higher privilege (ring 3 to 0) which permits system calls to run privileged instructions.

A system call number decouples locations in memory of the kernel from user space and allows each to evolve code independently.

6.) (20 points) In project #2, another way to handle signal delivery and signal completion is to have the kernel set the EIP to a **standard** signal processing function in user space (in libc) and have that libc function call the user's signal handler and then return to the kernel upon completion.

a) How would the Signal function and system call need to change to handle this model?

Solution 1: No change to signal

Solution 2: put the table of handers in user space and signal has a user space code.

b) How would setup frame need to be changed for this?

Solution 1: setup frame also pushes the address of the handler to be called by the generic function

Solution 2: pushes the signal number so the generic function knows which one to call

c) How would Sys_RegDeliver need to change?

Both solutions, no longer require Sys_RegDeliver since it was designed to provide the address of a function that is no longer needed, the standard libc function will return to kernel space. However, we need a way to register the address of the standard libc signal handler and Sys_RegDeliver could be used for this.