Problem Set 2

Due at the end of Unit 2: ___________________________

Name: ____

Please type your answers and submit as a PDF. Use the format <lastname>_<firstname>_PS2.pdf **as your filename.**

The problem sets ask questions about the important concepts of the reading and lecture materials. Each question should be answered in 1-4 paragraphs (no more 150 words).

We are looking for answers in your own words, NOT copies of the explanation given online, in the book, or in lecture material. Please explain your answers as if we are sitting together at a table discussing computer science concepts.

1. (5 pt) What are different ways to have shared memory between processes or threads? (describe each)
2. (5 pt) What problems can occur when using shared memory between threads?
3. (10 pt) How can the shared memory problems be solved using synchronization?
4. (5 pt) What is a semaphore? (describe and give an example of its use)
5. (5 pt) Describe the differences between conditional variable and monitor? (describe each, what is common, what is different, how is each used)
6. (5 pt) Describe the three classical problems cover that need synchronization?
7. (5 pt) What are the conditions needed for deadlock? (describe the condition, give an example)
8. (5 pt) Describe the following deadlock concepts: detection, prevention, recovery.
9. (5 pt) Describe the different performance criteria for scheduling.
   Describe each term, how it is calculated, and where you would might use each of the measurements.
10. (5 pt) Briefly describe each of the scheduling policies covered. (one or two sentences each)
11. (10 pt) Describe the scheduling algorithm used by Linux.

12. (10 pt) Given the code below answer the following questions (explain your reasoning):
    a. Is the function thread safe?
    b. Is the function reentrant?

```
int temp;
void swap(int *y, int *z)
{
      int local;

      local = temp; temp = *y;
      *y = *z;
      *z = temp; temp = local;
}
```

13. (25 pt) Using the process information below, create timing charts for the given policy. The values are all in number of ticks and you can assume no switching overhead. The multi-tasking timeline should assume a 20 tick time block. The IO column indicates when within the process an IO request is made and how long the wait for IO completion will take.

Process	**Original Priority**	**Arrival Time**	**Execution Pattern ticks <ticks IO blocked>**	**Deadline**
P1	**3**	**10**	**70 <20> 30**	**150**
P2	**2**	**70**	**10 <40> 10 <20> 20**	**220**
P3	**5**	**20**	**50 <10> 10 <20> 10**	**120**

Create a Gantt chart for each policy. Be sure to include the order of the queue at each decision point in time. Ignore information that is not used for the policy being used.

Using the Gantt charts below, fill out the table with calculations for each of the given performance measurements.

Policy	Execution Time	Wait Time	Turnaround Time	Response Time
FCFS				
SJF - preemptive				
EDF				

Problem Set 2

FCFS

When determining the queue order, if the currently running process is not blocked, move it to the end of queue when it completes its timeslice.

Queue																								
time	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200	210	220	230
CPU																								

SJF - preemptive

Queue																								
time	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200	210	220	230
CPU																								

EDF

Queue																								
time	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200	210	220	230
CPU																								