CS 4760	**Operating Systems**	**Test 2**
Name:	Fall 2017	Max Pts: 55

Important: This is an open book test. You can use any books, notes, or paper but no electronic device. *Do not log into the computer during the test, or use any electronic or communications device. Change your cell phones to silent mode*. Any calculations and rough work can be done on the back side of the test pages. If there is a syntax error in any program segment, just write it down and you will get full credit for the problem. Please write legibly; if I cannot read what you wrote, I'll give you a zero. You will lose five points for not writing your name.

1. [12 pt] Assume you have the following jobs to execute with one processor:

Process	Burst time	Arrival time
p_0	4	0
p_1	6	3
p_2	5	6
p_3	5	9
p_4	3	12

Calculate and show the average wait time for this set of processes using the following algorithms. Specify the arbitration rule used for each algorithm, if needed. Assume no time taken by the scheduler except where context switch cost is mentioned. Show your work.

(a) First in first out

(b) Shortest remaining time next (pre-emptive)

(c) Round robin, with a quantum of 3

(d) Round robin, with a quantum of 5 plus context switch time of 1 and 1 quantum used for initial scheduling

2. [6 pt] Unix/Linux shell is typically created as a user application; yet we classify the shell as a *special* application. What is the property of the shell that makes it different from a normal application?

3. [6 pt] Why do we need process groups?

4. [6 pt] What is the difference between a blocked state and a deadlock state?

5. [15 pt] You have a physical memory of 128MB, starting at address 0. Your operating system requires at least 14MB all the time. Consider the arrival of processes as follows:

Process	Burst time	Arrival time	Memory needed
p_0	5	0	59MB
p_1	19	4	109MB
p_2	22	8	3MB
p_3	15	8	98MB
p_4	12	11	113MB
p_5	14	17	57MB
p_6	25	22	104MB
p_7	9	24	37MB

Show the layout of memory, using first-fit algorithm, at times 10, 20, 30, 40, and 50.

6. [10 pt] In the code below, three processes are competing for six resources labeled `A` through `F`.

```
void p0()
{
  while ( 1 )
  {
    get ( A );
    get ( B );
    get ( C );
    crit_sec ( A, B, C );
    release ( A );
    release ( B );
    release ( C );
  }
}
```

```
void p1()
{
  while ( 1 )
  {
    get ( D );
    get ( E );
    get ( B );
    crit_sec ( B, D, E );
    release ( D );
    release ( E );
    release ( B );
  }
}
```

```
void p2()
{
  while ( 1 )
  {
    get ( C );
    get ( F );
    get ( D );
    crit_sec ( C, D, F );
    release ( C );
    release ( F );
    release ( D );
  }
}
```

Use a process resource graph to show the possibility of a deadlock in this implementation.