Name:

Operating Systems
Fall 2013
Test 1
October 2, 2013

Closed books, notes, cell phones, PDAs, iPods, laptops, etc. No headphones, please. No calculator needed.

You have 75 minutes to solve 6 problems. You get 10 points for writing your name on the top of this page. As with any exam, you should read through the questions first and start with those that you are most comfortable with. If you believe that you cannot answer a question without making some assumptions, state those assumptions in your answer.

Partial credit will be offered only for meaningful progress towards solving the problems.

Please read and sign below if you agree with the following statement:

In recognition of and in the spirit of the academic code of honor, I certify that I will neither give nor receive unpermitted aid on this exam.

Signature:________________________________

0	/10
1	/15
2	/15
3	/15
4	/10
5	/15
6	/10
Total	**/90**

1. (15 points) **Short Answer Questions**:

   a. A Control/Status register that contains the address of the next instruction to be fetched is called the _________.

   b. The __________ is a layer of software between the applications and the computer hardware that supports applications and utilities.

   c. _________ refers to the ability of an OS to support multiple, concurrent paths of execution within a single process.

   d. __________ was designed to keep the processor and I/O devices, including storage devices, simultaneously busy to achieve maximum efficiency.

   e. A situation in which multiple threads or processes read and write a shared data item and the final result depends on the relative timing of their execution is a _________.

   f. A significant point about the ___________ is that it contains sufficient information so that it is possible to interrupt a running process and later resume execution as if the interruption had not occurred.

   g. _________ is a section of code within a process that requires access to shared resources and that must not be executed while another process is in a corresponding section of code.

   h. The principal disadvantage of _________ is that the transfer of control from one thread to another within the same process requires a mode switch to the kernel.

   i. A situation in which a runnable process is overlooked indefinitely by the scheduler, although it is able to proceed, is __________ .

   j. The requirement that when one process is in a critical section that accesses shared resources, no other process may be in a critical section that accesses any of those shared resources is _________ .

2. (15 points) **Short attention span:** Answer the following questions (3 points each) on general concepts in operating systems.

   a. What is the purpose of system calls, and how do system calls relate to the OS and to the concept of dual-mode (kernel-mode and user-mode) operations?

   b. List two disadvantages of ULTs compared to KLTs.

   c. Which of the following instructions should be privileged?
      1. Read the clock
      2. Set value of timer
      3. Turn off interrupts
      4. Switch from user to kernel mode
      5. Clear memory

   d. What are the possible outputs of the following program, assuming the fork succeeds?

```
main()
{
     int pid;
     pid = fork();
     printf("%d \n", pid);
}
```

   e. What is the difference between multiprogramming and timesharing?

3. (15 points) **Process states:** The figure below contains seven states. In principle, one could draw a transition between any two states, for a total of 42 different transitions.
    a. Give an example of what could cause each of the possible transitions.
    b. List the impossible transitions and explain why they are impossible.

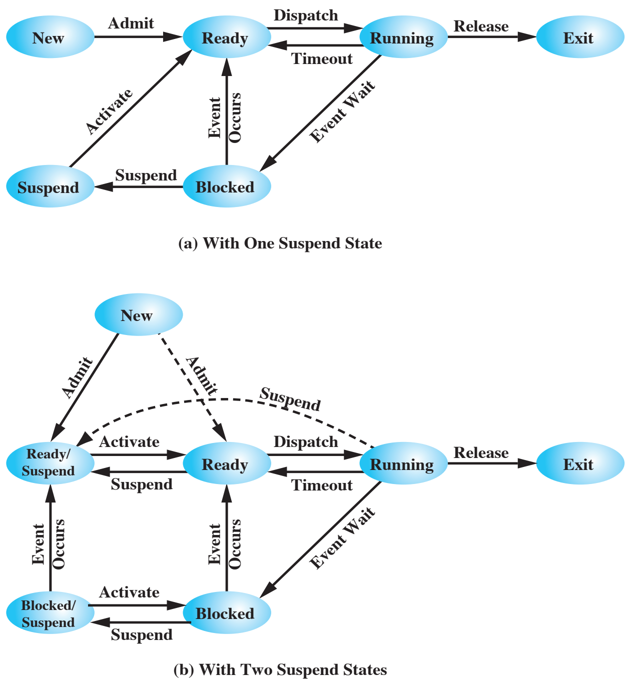

Figure 3.9 Process State Transition Diagram with Suspend States

4. (10 points) Can the following code lead to a race condition if executed in two threads of the same process if the threads are ULT? What if the threads are KLT? Explain your answer.

```
int x; //global

void undecided() //function run by each thread
{
     x = 10;
     while(1){
          x = x + 1;
          x = x - 1;
          if (x != 10)
               printf("x is %d", x);
     }
}
```

5. (15 points) The execution of the following program will create one or more processes. As a result, one or more statements will be printed to the screen. You may assume that the newly created processes are assigned PIDs 100, 200, 300, etc.
    a. Show the process hierarchy resulted from the execution of this program.
    b. What will this program print on screen when it executes?

```
#include <stdio.h>
main()
{
      int pid1, pid2, pid3;
      int counter;
      pid1 = 10; pid2 = 20; pid3 = 30; counter = 1;
      pid1 = fork();
      printf("%d, %d, %d, %d \n", pid1, pid2, pid3, counter);
      pid2 = fork();
      printf("%d, %d, %d, %d \n", pid1, pid2, pid3, counter);
      while (counter < 2){
     pid3 = fork();
     printf("%d, %d, %d, %d \n", pid1, pid2, pid3, counter);
     counter = counter +1;
}
```

6. (10 points) What is the output of the following code and why? Assume the program runs with argument 4.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
int fact; /* this data is shared by the thread(s) */

void *factorial(void *param); /* the thread */

main(int argc, char *argv[])
{
  pthread_t t1, t2; /* the thread identifier */
  if (argc != 2) {
    fprintf(stderr, "usage: a.out<integer value>\n");
    exit(0);
  }
  if (atoi(argv[1]) < 0) {
    fprintf(stderr, "%d must be >= 0\n",atoi(argv[1]));
    exit(0);
  }
  /* create the threads*/
  pthread_create(&t1,NULL,factorial,argv[1]);
  pthread_create(&t2,NULL,factorial,argv[1]);
  printf("created threads\n");
  pthread_join(t1,NULL);
  pthread_join(t2,NULL);
  printf("factorial = %d\n",fact);
}

/* The thread will begin control in this function */
void *factorial(void *param)
{
  int n =atoi(param);
  int i;
  fact = 1;
  if(n > 0) {
    for(i=1;i<= n; i++)
             fact = fact * i;
  }
  pthread_exit(0);
}
```