CS 346 – Review for Test 1
Here are some former exam questions.

1. Give a brief definition of each of the following OS terms.
   a. kernel

   b. shell

   c. system call

   d. context switch

   e. POSIX

   f. multiprogramming

   g. redirection

   h. device driver

   i. memory management

   j. process

   k. thread-join operation

2. Why is it not possible to implement an entire operating system in a high-level language?

3. An assembler is a program that translates assembly language instructions into machine code that can be directly interpreted by the hardware. Would an assembler be considered part of the OS kernel? Justify your answer.

4. Describe the different states that a process can be in during its lifetime.

5. What does this Linux command accomplish?

```
ls | grep 5 | wc
```

6. Describe an example of a program where creating multiple threads would make more sense than creating multiple processes.

7. Explain what an exec() system call does. When is the appropriate situation to invoke this call?

8. What information is shared among all the threads of the same process? What information can differ among each of the threads?

9. Consider the following abbreviated process table.

PID	PPID	NAME
7205	7199	bash
7325	7205	a.out
7326	7325	a.out
7327	7325	a.out
7329	7327	a.out
7330	7327	a.out

   a. Describe the hierarchical relationship among the 5 processes called a.out.

   b. Give a pseudo-code design of a C program using fork() calls in order to create child processes suggested by the process table. Assume that fork() returns 0 in the child, and a positive integer in the parent.

10. Processes P1 and P2 need to access a critical section of code. Consider the following synchronization construct used by the processes. Assume that the critical sections of code have very short execution times.

```
/* Process P1 */
while (true) {
  wants1 = true;
  while (wants2 == true)
    ;
  /* Critical section */
  wants1 = false;
  /* Non-critical code */
}
```

```
/* Process P2 */
while (true) {
  wants2 = true;
  while (wants1 == true)
    ;
  /* Critical section */
  wants2 = false;
  /* Non-critical code */
}
```

Here, wants1 and wants2 are shared variables, which are initialized to false. Answer the following questions about the above implementation of P1 and P2. Justify your answers.

a. Does it ensure mutual exclusion?

b. Is deadlock possible?

c. Is starvation (unbounded waiting) possible?

d. Must the two processes access their respective critical sections in strict alternation? (Strict alternation means that a process having just finished its critical section cannot re-enter it until the other process has entered and exited its critical section.)