CS 4760	**Operating Systems**	Test 1
Name:	Fall 2021	Max Pts: 62

Important: This is an open book test; you can use any books, notes, or paper. If there is a syntax error in any program segment, just write it down and you will get full credit for the problem.

1. [6 pt] What is the distinction between spatial locality and temporal locality?

   Spatial locality refers to the tendency of the code to access instructions and data that are in physical proximity. Temporal locality refers to instructions and data in locations that may not be physically close but are generally accessed together.

2. [6 pt] What is the role of `ioctl` in the OS kernel?

   `ioctl` is used to communicate control signals to the device as opposed to data transfer.

3. [10 pt] A computer has cache, main memory, and a disk used for virtual memory. If a referenced word is in the cache, 15ns are required to access it. If it is in main memory but not in the cache, 50ns are required to load it into the cache (this includes the time to originally check the cache), and then, the reference is started again. If the word is not in main memory, 11ms are required to fetch the word from disk, followed by 50ns to copy it to the cache, and then, the reference is started again. The cache hit ratio is 0.8 and the main memory hit ratio is 0.9. What is the average time in ns required to access a referenced word on this system?

   Fraction in cache: 0.8
   Fraction in main memory: (1 - 0.8) $\times$ 0.9 = 0.18
   Fraction read from disk: 0.02
   Average time: $0.8 \times 15 + 0.18 * (15 + 50) + 0.02 * (15 + 50 + 11000000)$
   $= 12 + 11.7 + 220001.3$
   $= 220025$

4. [10 pt] List and briefly explain five storage management responsibilities of a typical OS.

   (a) Process isolation: Should prevent the independent processes from interfering with the data and code segments of each other

   (b) Automatic allocation and management: Programs should be dynamically allocated across the memory depending on the availability (may or may not be contiguous)

   (c) Modular programming support: Programmers should be able to define program modules, and be able to dynamically create/destroy/alter the size of modules

   (d) Protection and access control: Different programs should be able to co-operate in sharing some memory space

   (e) Long-term storage: Users and applications may require means for storing information for extended periods of time

5. [6 pt] What is an instruction trace? What is the difference between the instruction trace for a single process and multiple processes executing on a uniprocessor?

   Instruction trace for a process is the sequence of instructions executed by the CPU for the process. For a single process, it is the sequence of instructions from that process. For multiple processes executing on a uniprocessor, the instruction trace is the interleaved instructions from different processes as they are executed by the CPU.

6. [12 pt] The following state transition table is a simplified model of process management, with the labels representing transitions between states of READY, RUN, BLOCKED, and NONRESIDENT.

	READY	RUN	BLOCKED	NONRESIDENT
READY	-	1	-	5
RUN	2	-	3	-
BLOCKED	4	-	-	6

   Interpret transition 2 as the fact that the process can change from RUN to READY. Give an example of an event that can cause each of the above transitions. Draw a diagram if that helps.

   1. Process is dispatched, or assigned the CPU

   2. A running process has finished its assigned quantum; timer interrupt

   3. A running process asks for a resource that cannot be immediately granted

   4. A waiting process is granted the resource

   5. A ready process is swapped out to make room for other processes

   6. A blocked process is swapped out to make room for other processes

7. [6 pt] I wrote the following code to create a child to do something and return. The return value is to be caught by the parent. Can you see any problem with the code that I wrote? How will you fix it?

```
int status;
pid_t pid = fork();
if ( pid < 0 )
    exit ( 1 );
if ( pid == 0 )
    wait(&status);
exit ( 0 );
```

   In the code, the child starts to wait while the parent exits. The child will never receive the signal it is waiting for. The fix is:

```
int status;
pid_t pid = fork();
if ( pid < 0 )
    exit ( 1 );
if ( pid > 0 )
    wait(&status);
exit ( 0 );
```

8. [8 pt] What is the difference between the semantics of wait/signal operations in case of semaphores and condition variables?

   Condition wait suspends the process while semaphore wait may or may not suspend the process depending on semaphore count. Condition signal resumes one suspended process and has no effect if there is no waiting process; semaphore signal increments the semaphore count and restarts a waiting process.