1. **True/False.** Circle the appropriate choice (there are no trick questions).
   
   (a) **T F** The two primary purposes of an operating system are to manage the resources of the computer and to provide a convenient interface to the hardware for programmers.
   
   (b) **T F** When the CPU is in kernel mode, the CPU is prevented from accessing areas of memory that are not owned by the current process.
   
   (c) **T F** An interrupt table contains the addresses of the handlers for the various interrupts.
   
   (d) **T F** If N batch jobs are scheduled using Shortest Job First scheduling, the total time it takes to execute the N processes is less than for any other batch schedule (assuming no two jobs take the same time).
   
   (e) **T F** Each thread of a process has its own virtual address space.
   
   (f) **T F** Every time a clock interrupt occurs, a context switch from one process to another is performed.
   
   (g) **T F** In a multiprogrammed system using partitioning (i.e. each process occupies a contiguous block of memory), addresses can be relocated at run time using base registers.
   
   (h) **T F** In a multiprogrammed system using partitioning, the Best Fit strategy (where a process is placed in the smallest hole in memory large enough for the process) provides the most effective use of memory.
   
   (i) **T F** In a virtual memory system, a virtual page and a physical page frame must be the same size.
   
   (j) **T F** In a virtual memory system, a virtual address and a physical address must be the same size.

   **Put the rest of your answers in the blue book**

2. (a) What is the difference between a trap and an interrupt?
   
   (b) What does the CPU hardware do when a trap or interrupt occurs? Describe the specific steps that the hardware performs (but just the hardware - not the OS or any process).
   
   (c) Why is a special instruction, e.g. IRET, for returning from an interrupt handler required in order to resume executing a user-level process. That is, why can’t a normal return instruction, e.g. RET, be used?
   
   (d) When a process executes a fork() system call, a duplicate process (i.e. the child process) is created. How does the code in the processes – since it is identical in both the parent and child processes – know which process is the parent and which is the child?
   
   (e) Give a simple example of code that would operate differently in the parent and in the child.

   **Please turn this page over.**
3. (a) Give a simple example of a race condition by writing a code segment (in C) which, if it were executed by two processes executed at roughly the same time, could cause a race condition. Explain what bug could arise from the race condition.

(b) Add some additional code, in conventional C (i.e. no semaphores or TSL), so that the race condition bug no longer exists.

(c) What is the disadvantage of the above solution compared to using TSL (test-and-set-lock)?

(d) Suppose C provided a procedure \texttt{tsl}(x) that atomically returns the value of \(x\) and sets \(x\) to 1. Modify your original code (from part (a)) to use \texttt{tsl} to fix the race condition.

(e) What is the disadvantage of your solution using \texttt{tsl} compared to using semaphores?

4. (a) Describe the lottery scheduling algorithm.

(b) What is the advantage of using lottery scheduling over strict priority scheduling (where the highest priority ready process is the next process chosen to run)?

(c) Given two processes in the READY state, one that is CPU-bound and one that is I/O-bound, which process should be given a higher priority for running next (all other things being equal)? Justify your answer.

(d) In the first programming assignment, you may have noticed that some processes were able to run for more than the quantum (which was 40ms) without being put back on the ready queue. Under what circumstances would this occur?

5. Suppose you have a virtual memory system where addresses are 22 bits and the page size is 4096 (i.e. \(2^{12}\)) bytes.

(a) How many bits of a virtual address are used to determine the virtual page number and how many bits are used to determine the offset?

(b) How many elements would a page table need to have?

(c) Suppose, during the execution of a process, the MMU performs the following virtual address to physical address translations:

\[
\begin{align*}
0000001011010110101111 & \rightarrow 00001011010101101111 \\
0000111110000010110100 & \rightarrow 00100000100100101110 \\
001010001010100010010 & \rightarrow 00000001010101001010 \\
000000101010000010010 & \rightarrow 000010111000000010010
\end{align*}
\]

How many different elements of the page table are accessed by the MMU to produce the above translations? Which elements of the page table are they and what page frame numbers do these elements contain?

(d) Briefly describe the sequence of events that occur when a process tries to access a location in its virtual address space that is not currently in RAM. Start with the initial attempt to access the location and end with the location being accessed successfully.