1. [20 points] A variant on the usual signal and wait semaphore operations are the \texttt{SignalMult} and \texttt{WaitMult} operations, with the following semantics:

\[
\text{WaitMult}(\text{sem, amt}) \text{ blocks until } \text{sem} > \text{amt} \text{ holds and}
\]
\[
\text{then indivisibly executes } \text{sem} := \text{sem} – \text{amt}.
\]

\[
\text{SignalMult}(\text{sem, amt}) \text{ indivisibly executes } \text{sem} := \text{sem} + \text{amt}.
\]

Consider a system of \( N \) processes that are accessing a shared database. At most \( R \) of these are readers, and at most \( W \) are writers. The safety property of interest is:

\textbf{Safety Property:} Any number of readers is allowed concurrent access to the database provided there are no writers accessing the database; a single writer is allowed access provided there are no readers and no writers accessing the database.

Give protocols that use \texttt{WaitMult} and \texttt{SignalMult} as the sole means of synchronization in an implementation of \texttt{StartRead}, \texttt{EndRead}, \texttt{StartWrite}, \texttt{EndWrite} operations that programmers can use to bracket sections of code that access the database and be guaranteed that the above safety property holds. Ignore questions of liveness or starvation. Protocols that are unnecessarily complex and/or do not exploit the full power of \texttt{WaitMult} and \texttt{SignalMult} will be penalized.
2. [20 points] For each of the following statements, either agree or disagree and explain your reason briefly, in at most three sentences (not an essay!).

a. If a disk driver is permitted to arbitrarily reorder disk I/O operations from a Unix (UFS) filesystem to minimize seek time and a crash occurs while the disk is being updated, then upon rebooting we might find that a single disk block is shown as being in two or more files.

b. Filesystem defragmenters rearrange data blocks on disk in order to ensure that contiguous blocks of a given file are physically contiguous on disk. This approach no longer works well because high rotational speeds of modern disks, coupled with relatively significant per-block processing latencies, imply that the optimal physical layout for sequentially accessed blocks is no longer physically contiguous. (Note for wizards: assume that the disk controller only supports block-at-a-time (cooked) accesses for a single, fixed block size, and does not support multi-sector or full-track accesses)

c. On a Unix-like filesystem with direct and indirect i-nodes (metadata blocks that hold the locations of data blocks), with no bad blocks on disk, and with a cold file cache, a researcher measures the average access time for each block of various N-block files, where she varies N from 1 to 20. The resulting graph appears on the right hand side. The increase in per block access time at 12 blocks is most likely due to a bad placement decision by the filesystem.

d. The popular CPU scheduling algorithm called “preemptive round robin” is a particularly good choice (assuming a very small choice for a quantum) for specialized applications subject to real-time constraints, such as the controller for a robot, in which motor commands must be issued “on time” or the robot might be damaged.

e. The storage requirements for an inverted page table containing mappings for N physical pages for M concurrent processes are O(M log N).

f. If we wish to guarantee the very best possible response time for interactive applications, we’ll need to use a scheduling policy that might not give the shortest possible completion times for long-running tasks.

g. Recall that mail servers deliver mail by appending incoming mail to files belonging to individual users. A stateless file system is ideally suited to act as the backing store for a mail spool that is shared by two replicated, load-sharing mail servers, as it can recover quickly from the crashes of either or both of the two mail servers or the fileserver itself.

h. The term “livelock” refers to a condition in which a paging algorithm is short on memory and makes poor page-out choices. Pages are paged in and out repeatedly, incurring high overhead.

i. One problem with heavy use of threads for parallelism and dynamically linked files for sharing is that an application can end up with a large, fragmented, and sparse virtual address space.

j. Large, fragmented and sparse virtual address spaces diminish performance by reducing spatial locality and consequently lowering cache hit ratios.
3. The following two code fragments employ TCP sockets in order to send a series of strings from a server to a client over an IP network. Assume that neither the server nor the client hosts fail, and that the code runs to completion. Answer the following questions, stating your assumptions clearly and completely:

/******************** SERVER CODE SNIPPET ****************/
char str_ptr[MAX_LEN]; /* MAX_LEN is longer than any string */
int str_len;
int sock_tcp;
/* Assume that the following subroutine sets up a connected TCP
* socket, and returns a valid socket ID */
sock_tcp = establish_socket(client);
while(str_len = read_next_string(str_ptr)) {
    /* read_next_string() returns the length of the string, and writes
     * the string into the array str_ptr. If no string, it returns 0. */
    /* Assume for this question that the return value of the following
     * send always equals str_len. */
    if (send(sock_tcp, str_ptr, str_len, 0) != str_len)
        die_with_error();
}
close(sock_tcp);

/******************** CLIENT CODE SNIPPET ****************/
char str_ptr[MAX_LEN+1]; /* MAX_LEN is longer than any string */
int str_len;
int sock_tcp;
/* Assume that the following subroutine sets up a connected TCP
* socket, and returns a valid socket ID */
sock_tcp = establish_socket(server);
while(str_len = recv(sock_tcp, str_ptr, MAX_LEN, 0) > 0) {
    /* write_next_string() provides the string of length str_len
     * to the application. */
    write_next_string(str_ptr, str_len);
}
close(sock_tcp);

a. [10 points] Does the client receive all of the bytes transmitted by the sender? Why or why not?

b. [10 points] Is the client guaranteed to receive the strings in tact? In other words, for every string read by the server's read_next_string() routine, is the same string and string length written by the client's write_next_string() routine? Why or why not?
4. Relational Model

a. [7 points] Consider the following relations:

- Parts(pid, pname, color)
- Catalog(pid, sid, price)
- Suppliers(sid, sname, address)

Write the following query in SQL: Find all the suppliers that supply the second largest number of parts. In case there is no such supplier, your output should be empty.

b. Consider a relation R(A,B,C,D,E,F) with the following functional dependencies:

- A -> BC
- CD -> E
- B -> D
- E -> A

   i. [5 points] List all the keys for R.
   
   ii. [5 points] What is the highest normal form that R is in (1NF, 3NF, or BCNF)? Justify your answer.

c. [6 points] Assume that you are given relation R(P,Q,R,S,T,U). Assume that there are two separate unclustered B+-tree indices, one on R.P and one on R.Q. Consider the following query:

\[
\text{SELECT DISTINCT R.P, R.Q} \\
\text{FROM R}
\]

Describe three reasonable choices (whose cost depends only on the number of pages of R, and not on the number of tuples in R) for processing this query, in one sentence each.
5. a. [5 points] State-of-the-art recovery algorithms such as ARIES assume strict two-phase locking. Clearly explain what aspect of recovery is simplified by assuming strict two-phase locking (as opposed to simply using regular two-phase locking). Illustrate the problem with a transaction schedule that is allowed by two-phase locking but not strict two-phase locking.

b. [12 points] Assume that you want to design a recovery algorithm that allows two-phase locking instead of strict two-phase locking (for extra concurrency).

   i. What additional processing would you have to do at transaction commit time? Clearly state the extra steps involved in addition to the normal commit processing.

   ii. What additional processing would you have to do at transaction abort time? Clearly state the extra steps involved in addition to normal abort processing.
6. Consider the Q666, a machine with:
   - 4 byte words
   - 64 bit virtual addresses
   - 48 bit physical addresses
   - 8192 byte (8KB) page size
   - A 128KByte physically indexed, physically tagged, direct-mapped cache with a block size of 64 bytes

a. What are the widths of the following fields of the address?
   - Virtual page number
   - Page frame number (a.k.a. physical page number)
   - Cache tag
   - Cache index
   - Cache word offset

b. The Q665 is a similar machine, but has a different cache organization. Its cache is only 128 bytes, and uses 32 byte blocks.

The following two C programs are executed on the Q665:

/* program A */
for (i=0; i<4; i++)
  for (j=0; j<2; j++)
    a[j][i]++;

/* program B */
for (j=0; j<2; j++)
  for (i=0; i<4; i++)
    a[j][i]++;

For both programs, a[][] is a 32 × 32 integer array, where an integer corresponds to one 4-byte word.

Calculate the miss rates for both programs as well as the final contents of the cache. Show the data and tags for each line as well as the valid bit; label the words in the cache with the array indices for a, leaving unknown regions of the cache blank. Assume the cache is write-through. Assume that the physical address of a[0][0] is 0x420, and that all the valid bits in the cache are initially 0.

c. Assume that the Q665 has a 1.5GHz clock rate, a cache hit time of 2 cycles, and a memory miss penalty of 120ns. Calculate the average memory access time (in cycles) for programs A and B.