Consider the short (and useless) C program off to the right. Each of the variables \{a, b, c\} references an integer value that occupies some memory at runtime. Answer the following questions for each of the integer values referenced by \{a, b, c\}.

Where is/are \{a, b, c\} stored in virtual memory? (i.e., what segment)

How/when is the virtual memory for \{a, b, c\} allocated? How/when is it released?

When the main thread accesses \{a, b, c\}, how is the virtual address computed or determined?

How/when is the machine memory for \{a, b, c\} allocated? How/when is it released?

When the main thread accesses \{a, b, c\}, how is the machine address computed or determined?
Part 2. Déjà vu all over again (60 points)

Implement monitors using semaphores. As always, “any kind of pseudocode is fine as long as its meaning is clear”. Don’t forget to implement the condition variables! You may omit broadcast/notifyAll, and you may assume that all threads using your implementation are “well-behaved”.
Part 3. Waiting for a core (60 points)

In general, threads in the Ready state (Runnable) are linked into a list called a ready queue or runqueue. (More precisely, their thread objects or Thread Control Blocks are linked into a list.) Answer the following questions about ready queues. Each question is asking for short phrases in the “lingo” of this class. You can explain a little, but please keep it brief.

(a) What events would cause the kernel to place a thread on the ready queue? List as many causes as you can.

(b) What events would cause the kernel to remove a thread from the ready queue? List as many causes as you can.

(c) Most modern kernels have multiple ready queues. Why is it useful to have multiple ready queues? List as many reasons as you can.

(d) If a thread is not on a ready queue, what other data structures in the kernel might point to it? List as many examples as you can.
Part 4. Brave new world (60 points)

Alice, Bob, and their friends interact via the social networking service Faith. They protect their communication using secure sockets (SSL) for transport-layer security. Nancy wants to spy on their communication. Assume that Nancy can subvert any network provider and so intercept and/or inject packets at any point in the network.

(a) Nancy cannot “break” the cryptography, but she can still spy on SSL-encrypted communication if she can obtain the right keys. Outline two approaches that Nancy might use to obtain useful keys. Be sure to explain how Nancy would use the keys obtained in each approach (if it succeeds).
Part 4. Brave new world (continued)

(b) Now suppose that Nancy also wants to modify the SSL-encrypted communication between Faith and her clients as it occurs, without detection by any of them. Can she do it using the approaches from (a)? Explain.

(c) Faith and other service providers are planning to enhance their security with a scheme called Perfect Forward Secrecy to defend against the following vulnerability of SSL: if Nancy obtains Faith's private key, then Nancy can decrypt all previous communication between Faith and her clients, assuming that Nancy has also intercepted and retained a copy of those network packets. Explain how Nancy would use the key to decrypt the saved traffic.
Part 5. Spinning in circles (60 points)

This part deals with performance and data safety for a data storage volume under various designs. Suppose that the system issues a stream of read and/or write requests for random block numbers in the volume. For example, these requests might result from some workload on a file system similar to your DeFiler (Lab #4).

In the baseline design the storage volume resides on a single disk drive (HDD). The following questions each propose a change to the baseline design, and ask you to state the effect of the change relative to the baseline design. You may assume that the request stream is continuous and balanced: the next request arrives for a disk just before its previous request completes. I am looking for answers that are specific and quantitative. You might need some other assumptions: if so, be sure to state them.

(a) If we double the rotational speed or arial density of the single disk, how does it affect throughput and the average per-request response time, other things being equal?

(b) If we double the block size, how does it affect throughput and the average per-request response time, other things being equal?

(c) Suppose we distribute the volume across N disks using striping (RAID-0). How does it affect throughput and the average per-request response time, other things being equal? How does it affect the probability of a failure that results in data loss?

(d) Suppose we distribute the volume across N disks using mirroring (RAID-1). How does it affect throughput and the average per-request response time, other things being equal? How does it affect the probability of a failure that results in data loss?

(e) Suppose the volume has N disks, and blocks are distributed across the disks using striping with parity (RAID-5). How does it affect throughput and the average per-request response time, other things being equal? How does it affect the probability of a failure that results in data loss?
Part 5. Spinning in circles (continued)

(f) Now suppose the system implements atomic commit for groups of writes using logging. How does it affect throughput, other things being equal?

(g) Now suppose the system implements atomic commit for groups of writes using shadowing. How does it affect throughput, other things being equal?

(h) Suppose the system caches blocks in memory, using a structure similar to your Lab #4 buffer cache. You may assume that the buffer cache can hold $M$ blocks out of a capacity of $C$ blocks for the disk volume, where $M < C$. How does adding the cache affect throughput and the average per-request response time, other things being equal?
Part 6. Extra credit

What is a thread? Illustrate with a picture. Extra extra points if you can explain it in a more entertaining way than I did. (Low bar?)