Midterm review
scope -- lectures, assignments, exercises, examples
strategies -- two pass process, my limits, etc.
samples

Some caveats
My exams are difficult.
Always curved.
Professionals (you know who you are) are not included in calculating curves.
Looks like a loooong exercise.
10 answers, 10 points per answer, points do not indicate difficulty.
Roughly 1/3 relatively straightforward questions.
Roughly 1/3 intermediate difficulty questions.
Roughly 1/3 "unfair:)" questions.
2-pass process
   Do problems you understand first.
Then do harder ones.
Budget time to check your answers for whether I am being sneaky.
Some strategic hints for open book exams

Index your notes -- with post-it notes -- so you can find things you need quickly. Review the examples -- I will not veer far away from them.

If you do this carefully, you'll find that you don't have to refer to the notes!
I suggest that you outline the answer first, and then write it nicely if you have more time. You will definitely get partial credit for the outline, but no credit for a blank.
Bounded buffers and pipes
Lock ordering
Banker's algorithm
Deadlock.
Race conditions - nondeterminism
Buddy system- fragmentation
TLB -- performance
resource to allocate

process that needs resource

P holds R

P wants R

Cycle in RAG ⇒ deadlock

P1

P2

R1

R2
Banker's algorithm example

Principles

- Runnable if deficits are 0
- Blocked otherwise.
- Safe if there is a completion schedule
- Unsafe if not.
- If requested != granted, you can decide in the schedule when to grant.
IA32:

  Hardware segment table.
  All loaded at once.
  (has a TLB that short-circuits segment, page addressing)

IA64:

  Hardware page table (not segment).
  Loaded one page at a time.
  There is no set architecture for segments.
  TLB is the only page cache.

A segment is a set of pages, typically logically contiguous, with the same attributes: read/only; read/write; shared; etc.
A page is the smallest unit that an OS can allocate to a process.
Overview:
    Always allocate a power of two of bytes. 
    Largest block is $2^{13} = 8,192$ bytes. 
    When you ask for $m$ bytes, actually get $2^k$ bytes, 
    where $k$ is the smallest such that $m \leq 2^k - 16$ 
    We only store the free lists for each power of 2 
    between 5 and 13 (smallest allocation is 32 bytes = 16 
    bytes data, 16 bytes overhead).

Algorithm:
    Always allocate 8192 bytes 
    Split in half until you get to an appropriate size, and 
    link all unused fragments into free lists. 
    Then use one element from the split.

Free algorithm:
    Don't actually free anything. Just mark it unused.

Internal fragmentation: memory that is allocated, but 
technically unused by the process. 
Time/space tradeoff: using 2x the space leads to $O(1)$ 
malloc, $O(1)$ free. 
Corruption: descriptor for each fragment is stored next 
to fragment, subject to corruption.