True/False:
a. LRU caches are more complex than random caches.

True, random caches are easy to implement in hardware, LRU caches requires storing information about access time.

b. The optimal page replacement strategy is to evict the page which will be accessed least frequently on average.

False, it's to evict that which will be accessed furthest in the future, as we said in class. Can you think of how prove this to yourself?

c. Write-through caching is better when a small number of items are modified frequently.

False, write-back caching is better here, because you have a small "hot" working set.

Short Answer:
In class we described 4 types of cache misses. Name and describe two of the types we discussed.

Compulsory Misses: sad facts of life, e.g. program start up.
Conflict Misses: in not-fully associable caches.
Capacity Misses: if cache capacity is too small.
Coherence Misses: Caused by external processors or I/O devices!

Below is a pseudo-code for the clock page replacement algorithm (we are assuming the clock is entirely "full" with pages, and a new page must be found). Fill in the blank:

```
replaced = false;
while (!replaced) {
  1. if (current page use_bit==0) {replace page, replaced=true} else
{use_bit=0}


  2. Advance clock hand

}
```

Longer Answer:
Caching: Assume a computer system employing a cache, where the
access time to the main memory is 100 ns, and the access time to the cache is 20ns.

a. (2 points) Assume the cache hit rate is 95%. What is the average access time?
*Average Access Time = Hit*cache_access_time + (1-Hit)*memory_access_time*

*= 0.95*20 ns + 0.05*100 ns = 24 ns*
*Alternatively, we accepted solutions that included the cache time in the memory access time: AAT = 0.95 * 20 ns + 0.05 * (20 ns + 100 ns) = 25 ns.*

b. (2 points) Assume the system implements virtual memory using a two-level page table with no TLB, and assume the CPU loads a word X from main memory. Assume the cache hit rate for the page entries as well as for the data in memory is 95%. What is the average time it takes to load X?

*The Average Memory Access Time for X (AMAT) requires three memory accesses, two for each page entry, and one for reading X: 3*24 = 72 ns. The alternate solution from (a) yields 3*25 = 75 ns. We only accepted the alternate solution for (b) if you derived the same value for (a).*

Longest Answer (From Fall 2012 Midterm):

Assume a system with a two level page table. The virtual memory address space is 32 bits and the physical memory address space is 16 bits.

a. Make sure that each translation table fits in a page.

i. What is the optimal page size?

4KB - use 12 bits to keep page table entries on a single page

ii. Specify the length of each field in the virtual address.

bits 0-11: offset (12 bits)
bits 12-21: vpn #2 (10 bits)
bits 22-31: vpn #1 (10 bits)

b. Assume you add to your system a 4-way set-associative data cache with 16 total cache blocks. Each block in the cache holds 8 bytes of data. In order to address a specific byte of data, you will have to split the address into the cache tag, cache index and byte select. Assume there are no modifiers bits in the table.

i. How many bits are used for byte selection? *3 (8 bytes of data = 2^3)*

ii. How many bits are used for the cache index? *2 (16 blocks / 4 per set = 4 sets = 2^2)*

iii. How many bits are used for the cache tag? *11 (16 - 3 - 2)*