COMP 530 Quiz #1

My name is (print **CLEARLY**) _____

Circle the **best** answer for each question.

(1) A parity disk is **not** useful for which of the following?

(a) Recovering from the failure of a particular disk
(b) Increasing the write throughput of the system
(c) Recovering from the failure of two disks simultaneously
(d) Both (b) and (c)

(2) Imagine that I use a parity disk for recovery, and my algorithm works by ensuring all sums are even. I have four disks plus a parity disk. For a page, the bits stored on my parity disk are 0110. The bits on disk one are 1010, the bits on disk two are 1111, the bits on disk three are 1011, and disk four crashes. What are the bits for that page on disk four?

(a) 0110
(b) 1001
(c) 0111
(d) 1000
(e) None of the above

(3) It I put all of the parity bits on a single parity disk, what problem might I expect?

(a) Write latency will be high, since all writes will have to hit the parity disk
(b) If the parity disk crashes, I won't be able to recover its contents
(c) Read latency will be high, since all reads will have to hit the parity disk
(d) Both (a) and (b) are potential problems

(4) What is a disadvantage of using mirroring (RAID 1) as a way to facilitate recovery from hard disk failures?

(a) Mirroring reduces the read throughput by a factor of two
(b) You have to reformat the disks any time there is a failure
(c) You have to buy twice as many disks in order to recover from a single-disk failure
(d) There is no way to keep copies of the same data consistent across two disks

(5) What is the "classic" bad case for the LRU algorithm?

(a) When an application repeatedly scans a file slightly larger than the buffer size
(b) When an application has a small set of pages that it accesses frequently
(c) When an application repeatedly scans a file slightly smaller than the buffer size
(d) When an application alternates reads with writes

(6) Why might we expect the clock algorithm to be something of an approximation to the LRU algorithm?

(a) Unlike other buffering algorithms, the clock algorithm can handle repeated scans over a file that is slightly larger than the buffer size
(b) Because there is no notion of “time” in the LRU algorithm
(c) Since a page is kicked out of the buffer only when the clock arm points to it and it has been accessed recently, older pages tend to be used to replace never pages
(d) When a page is accessed it can’t be kicked out of the buffer for awhile (at least one full rotation of the clock arm), and so older pages tend to be evicted.

(7) What is the one clear advantage of a hard disk versus a SSD (solid state drive)?

(a) It costs less to facilitate a certain number of bytes transferred from disk to RAM per second using hard disk
(b) The hard disk typically lasts longer before it fails
(c) It costs less to facilitate a certain number of “seeks” or random lookups per second using hard disk
(d) The hard disk costs less per byte of storage

(8) Why might the clock algorithm allow a more efficient implementation than the LRU algorithm?

(a) The clock algorithm uses less buffer memory per page than the LRU algorithm
(b) The clock algorithm does not need to do anything when a page in the buffer is touched
(c) The clock algorithm does not require a data structure update every time an older page it touched
(d) The clock algorithm can be used by multiple threads simultaneously, and LRU cannot

(9) Under what circumstances should a page that has been evicted from a buffer have its contents written to disk?

(a) Always
(b) If the version of the page in RAM has been written to since the time it was loaded into RAM from disk
(c) If the disk has failed since the page was loaded into RAM from disk
(d) If the database system has failed since the page was loaded into RAM from disk