Version A

Exam #2

CSE 3320.002

Fall 2014

Name: _____

UTA ID:_____

"I certify that the following work is my work alone and I will follow the highest standards of integrity and uphold the spirit of the Honor Code"

Signature:_____

Directions: This is a closed book, closed notes exam. You may use a hand written 8.5 x 11 sheet of paper with notes. Please answer the questions briefly. Complete sentences are not necessary. Write your answers legibly. Unreadable answers will be counted wrong. You may write on back if needed. A table of the powers of 2 is available on the last two pages of the test.

1. (15pts.) Describe memory paging. Why do we use it? What are its benefits?

2. (10pts.)In a virtual memory environment with 2 GB addressable space, where pages are 2KB bytes in size:
   1. How many entries are in the page table (maximum)?
   2. How would 31-bit addresses be used (how many page bits, how many offset bits)?

3. (15pts.) Given a file system that uses inodes to represent files. Disk blocks are 32 KB in size, and a pointer to a disk block requires 4 bytes. This file system's index nodes have 12 direct disk blocks, as well as a single second level indirect disk block. What is the largest file that can be held using this in inode layout?

4. (15pts.) Given a page request reference string of D E F A B E A C D F and a page table size of three, calculate how many page faults will occur with the optimal page replacement algorithm. Assume pages A B C are initially loaded into the page table automatically. If all pages are equally replaceable pick the first available.

5. (15pts.) Given a page request reference string of D E F A B E A C D F and a page table size of three, calculate how many page faults will occur with the least recently used page replacement algorithm. Assume pages A B C are initially loaded into the page table automatically. If all pages are equally replaceable pick the first available.

6. (15pts.) Given the following request queue -- 85, 170, 24, 11, 42, 17, 1, 101 with the disk head initially at the track 90 initially moving in the negative direction. The beginning of the disk at 0 and the end of the disk is at 199. Calculate the travel time for the SCAN algorithm.

7. (15pts.) Given the following request queue -- 85, 170, 24, 11, 42, 17, 1, 101 with the disk head initially at the track 101, the beginning of the disk at 0 and the end of the disk at 199. Calculate the travel time for the C-LOOK algorithm. Assume reads are only made in the positive direction. The disk head is initially moving in the positive direction.

Extra Credit

1. (5 pts) Consider a system where free space is kept in a free-space list. Suppose that the pointer to the free-space list is lost. Can the system reconstruct the free-space list? Explain your answer.

Extra Credit

2. (5 pts) You have been made developer of an embedded system that monitors air quality. The system writes a small 32 byte file every 15 minutes. The file system used by your device has inode blocks defined as:

```
#define V2_NR_DZONES  9
typedef zone_t int32_t;

EXTERN struct inode {
  mode_t   i_mode;    /* file type, protection, etc. */
  nlink_t  i_nlinks;  /* how many links to this file */
  uid_t    i_uid;     /* user id of the file's owner */
  gid_t    i_gid;     /* group number */
  off_t    i_size;    /* current file size in bytes */
  time_t   i_atime;   /* time of last access (V2 only) */
  time_t   i_mtime;   /* when file data last changed */
  time_t   i_ctime;   /* when was inode itself changed */
  int32_t  i_blocks[V2_NR_TZONES]; /* data blocks */
     …
  <remainder of struct not saved on disk>
}
```

With out modifying your file system layout, changing the block size, or your data files how will you reduce the internal fragmentation caused by your data files and maximize the amount of sensor data that can be stored? Explain your approach. No code needed.

Version A

Powers of Two

Bytes	bits	bit	Byte	KB Kilobyte	MB Megabyte	GB Gigabyte
	2	1				
	2	2				
	2	4				
Bytes	bits	bit	Byte	KB Kilobyte	MB Megabyte	GB Gigabyte
2	2	8	1			
2	2	16	2			
2	2	32	4			
2	2	64	8			
2	2	128	16			
2	2	256	32			
2	2	512	64			
2	2	1,024	128			
2	2	2,048	256			
2	2	4,096	512			
Bytes	bits	bit	Byte	KB Kilobyte	MB Megabyte	GB Gigabyte
2	2	8,192	1,024	1		
2	2	16,384	2,048	2		
2	2	32,768	4,096	4		
2	2	65,536	8,192	8		
2	2	131,072	16,384	16		
2	2	262,144	32,768	32		
2	2	524,288	65,536	64		
2	2	1,048,576	131,072	128		
2	2	2,097,152	262,144	256		
2	2	4,194,304	524,288	512		

Version A

Bytes	bits	bit	Byte	KB Kilobyte	MB Megabyte	GB Gigabyte
2	2	8,388,608	1,048,576	1,024	1	
2	2	16,777,216	2,097,152	2,048	2	
2	2	33,554,432	4,194,304	4,096	4	
2	2	67,108,864	8,388,608	8,192	8	
2	2	134,217,728	16,777,216	16,384	16	
2	2	268,435,456	33,554,432	32,768	32	
2	2	536,870,912	67,108,864	65,536	64	
2	2	1,073,741,824	134,217,728	131,072	128	
2	2	2,147,483,648	268,435,456	262,144	256	
2	2	4,294,967,296	536,870,912	524,288	512	
Bytes	bits	bit	Byte	KB Kilobyte	MB Megabyte	GB Gigabyte
2	2	8,589,934,592	1,073,741,824	1,048,576	1,024	1
2	2	17,179,869,184	2,147,483,648	2,097,152	2,048	2
2	2	34,359,738,368	4,294,967,296	4,194,304	4,096	4
2	2	68,719,476,736	8,589,934,592	8,388,608	8,192	8
2	2	137,438,953,472	17,179,869,184	16,777,216	16,384	16
2	2	274,877,906,944	34,359,738,368	33,554,432	32,768	32
2	2	549,755,813,888	68,719,476,736	67,108,864	65,536	64
2	2	1,099,511,627,776	137,438,953,472	134,217,728	131,072	128
2	2	2,199,023,255,552	274,877,906,944	268,435,456	262,144	256
2	2	4,398,046,511,104	549,755,813,888	536,870,912	524,288	512