CSCI 480 Name ____________________________
Spring, 2020

Test 2

Scores:

Part I	(18 points)	__________
Part II	(24 points)	__________
Part III	(16 points)	__________
Part IV	(32 points)	__________
Part V	(10 points)	__________
Total	(100 points)	__________

Part I. True or False (2 points each)

___ 1. When we use virtual memory, one way to cause thrashing is to assign only a few frames to a process with many pages.

___ 2. It is faster to read 1 byte from a hard disk than to read 100 bytes at 100 widely scattered addresses in memory.

___ 3. The number of bytes per page is usually a power of 2.

___ 4. The File Allocation Table is not stored in memory.

___ 5. A "dirty bit" is used to indicate whether the contents of a frame have been modified.

___ 6. The Translation Look-Aside Buffer is located in disk space.

___ 7. Segmented memory makes it easy for two processes to share a block of code (such as a library of standard functions).

___ 8. It is impossible for a single line of source code to cause a page fault.

___ 9. Information about the first cluster used to store a file is typically stored in a directory or in an index block.

Part II. Memory Management

10. (a) (2 points) Some events in a C++ program may require the dynamic allocation of some number of bytes of memory. Describe one of these.

(b) 2 points) A danger in some kinds of programming is a "memory leak". Give a brief example of C++ code that would create a memory leak.

11. (3 each) Suppose we are managing memory with paging. Here is a part of the page table. Assume that the size of a page or a frame is 2 kilobytes = 2048 bytes. The starting address of frame K is therefore K * 2 kilobytes.

Page Table:

Page	Frame
0	10
1	5
2	0
3	4

For each of these logical addresses, compute the page numbers, offset and physical address. (The numbers are all in base 10.)

	Logical Address	Page	Offset	Physical Address
(a)	1016	____	______	________________
(b)	5132	____	______	________________
(c)	3524	____	______	________________
(d)	6888	____	______	_______________

12. (2 each) Suppose we are managing memory with segments. Here is a part of the segment table. (The numbers are all in base 10.)

Segment Table:

Segment	Base	Length
0	4820	480
1	6148	98
2	2000	1720
3	2720	2100
4	5300	848

Examine each of these logical addresses. If the address is invalid, say so. Otherwise, calculate the corresponding physical address.

	Logical Address	Valid?	Physical Address (if valid)
(a)	2,1680	_____	________________
(b)	4,362	_____	________________
(c)	0,296	_____	________________
(d)	3,2478	_____	________________
(e)	1,72	_____	________________

Part III. Virtual Memory

13. (2 points) With well-managed virtual memory, a page that is seldom used will usually not be in a frame. What would be an example of code that we need in a program but will not often use?

14. (5 points each) Suppose we are trying to manage virtual memory. For each process, we have a sequence of pages it needs to use, and we have some number of frames available. We need to decide which pages go in which frames as the process runs.

(a) Suppose the process has 6 pages (0 to 5) and that there are 4 frames (0 to 3) available. The pages will be used in this order:

0 4 1 3 2 4 5 1 3 1 2 4

The frames are all initially empty. Use the First-In-First-Out algorithm and fill in the blanks in this chart to show that is in each frame at each point. Also indicate whether a page fault occurs (yes or no).

Be careful. Errors here will accumulate.

Page Used	Frame 0	Frame 1	Frame 2	Frame 2	Page Fault?
0	0	empty	empty	empty	Yes
4	0	4	empty	empty	Yes
1	0	4	1	empty	Yes
3	0	4	1	3	Yes
5					
4					
0					
1					
2					
4					
3					
5					

14 continued.

(b) Now redo part (a) but use the <u>Least-Recently-Used</u> algorithm instead.

Page Used	Frame 0	Frame 1	Frame 2	Frame 2	Page Fault?
0	0	empty	empty	empty	Yes
4	0	4	empty	empty	Yes
1	0	4	1	empty	Yes
3	0	4	1	3	Yes
2	2	4	1	3	Yes
4	2	4	1	3	No
5					
4					
0					
1					
2					
4					
3					
5					

15. (2 points) Suppose we are using virtual memory and we have 92 frames available for 4 processes. We could:

(a) deal out 23 frames to each process as a fixed scheme, or

(b) divide all of the frames among the processes in proportion to their sizes, or

(c) give frames to anyone who asks for one until we run out.

Pick <u>one</u> of these methods and name a disadvantage of it.

Part IV. Disks and File Systems

16. Suppose our operating system manages files with clusters of 4 kilobytes (= 4096 bytes) each and our file is 117,372 bytes long.

(a) (2 points) How many clusters will be needed to store this file? Show your work.

(b) (2 points) How many bytes are unused at the end of the last cluster? Show your work.

17. (2 points) Reading a file is fastest if the file is stored in contiguous disk space. Give a reason why it is not always practical to store files in this way.

18. (2 points) Name two kinds of metadata about a file which we might find in a directory listing or an index node.

19. (2 points) One of the ways to manage the collection of clusters assigned to a file is the "linked-list allocation" method. Describe a disadvantage of this method.

20. (2 each) Suppose we want to use "FAT-16" as our file system.

(a) How many clusters do we have?

(b) How much space is needed for the entire File Allocation Table (in bytes)?

(c) If our disk has a capacity of 42 gigabytes, how large is each cluster (in bytes)?

(d) If I use a bit vector to keep track of free clusters, how large is the bit vector (in bytes)?

21. (4 each) Suppose we are scheduling the operation of a hard disk. At the moment, there are 5 processes in a queue requesting read operations, each for small amounts of data. The tracks containing the data they want to read are:

42 (for the process at the top of the queue),
71,
8,
57 and
13 (for the process at the back of the queue).

Moving from one track to the next takes about 0.8 millisecond. Assume the read/write head move one track at a time. The Read/Write heads are presently at track 6.

For each of these algorithms, what is the order in which we will access the tracks, and how much time will we spend moving track to track in all?

(a) First-Come-First-Served

(b) Shortest-Seek-Time-First

22. (2 points each) Suppose a hard disk has a seek time of 11 milliseconds and a data transfer rate (disk to main memory) of 280 million bits per second. We want to read all of a file which is 11.2 million bytes long. The file is stored on 134 contiguous tracks, and the time to move track-to-next-track is 0.8 millisecond.

Show your work in answering these:

(a) How much time will we spend moving track-to-track as we read the file (in milliseconds)?

(b) How long will it take to transfer all the data (in milliseconds)?

(c) What is the total time to read the file (in milliseconds)?

Part V. Coding.

23. (10 points) Suppose we are extending the microshell program we invented in Assignment 3. We want to be able to handle a command-line pipe, as in:

```
myprompt>  ProgramA || ProgramB
```

where "||" is our symbol for output redirection.

Assume we have already parsed the line. Assume ProgramA and ProgramB have no command-line arguments.

We will use pipe(D) in the microshell, where D is an array of two integers. After that we will use fork() twice in the microshell to create two child processes C1 and C2. (We will ignore the possibility that fork() might fail.)

We intend to execute ProgramA in C1 and ProgramB in C2.

We need to arrange for the standard output of ProgramA to go into the Write end of the pipe and for the standard input of ProgramB to come from the Read end of the pipe.

You may assume we have:

```
pipe(D);
M = fork();
if (M == 0)
 {
  code for C1
 }
else
 {
  N = fork();
  if (N == 0)
   {
    code for C2
   }
  else
   {
    code for the microshell
   }
 }
```

Here (out of order) are some lines of code we might need:

Rearrange the lines to write the code. You do not need to fill in all the arguments for execvp().

23 continued.

(a) (4 points) Assume we have done the first fork() and we are writing code for C1. What code do we need?

(b) (4 points) Assume we have done the second fork() and we are writing code for C2. What code do we need?

(c) (2 points) What code do we need in the microshell after these?