Question 1. *(10 points) Describe the difference between internal and external fragmentation. What effect does paging have on each type of fragmentation?*

Question 2. *(10 points) Explain why systems using paging usually choose a page size that is a power of 2 (e.g., $2^8 = 256$ bytes, $2^9 = 512$ bytes, etc). What is the disadvantage of choosing a page size that is not a power of 2?*

Question 3. *(20 points) Consider a system with 32 byte pages and a total memory size of 2048 bytes. Assume that the system can access individual 4-byte words as the smallest unit of memory addressing (i.e., as in a typical 32-bit system). Recall that this means that an offset of 1 into a page means the second 4-byte word of the page (i.e., bytes 4-7).*

1. *(5 points) What is the total number of addressable words supported by this memory? How many different pages can be supported?*

2. *(5 points) How many bits are needed for an address? Of these, how many bits are needed for the page number (p) and how many for the offset (d)?*

3. *(5 points) Assuming the (partial) page table shown below, translate virtual address "28" to a physical address (i.e., the kth word of physical memory).*

Page	Frame
0	5
1	14
2	9
3	7
4	18
...	...

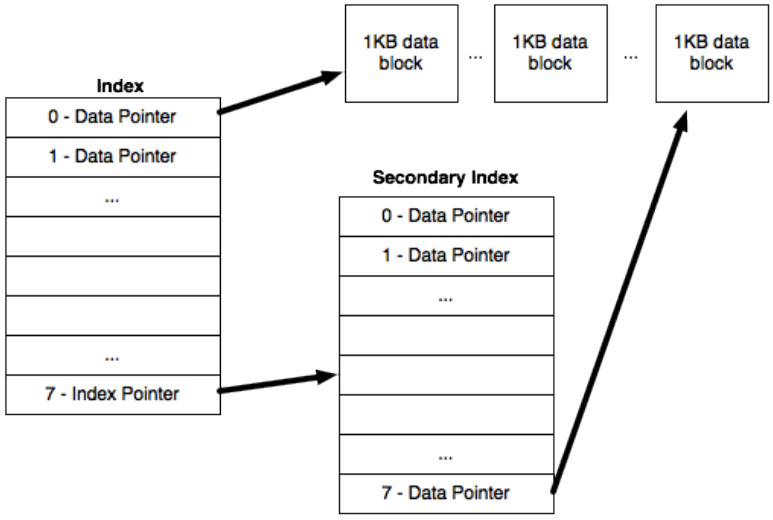
4. *(5 points) Suppose you extend your paging system to support segmented paging, where each process will have 7 segments. How many bits will be needed to encode a virtual address? How many bits for a physical address?*

Question 4. *(15 points) Determine how the FIFO and MIN page replacement algorithms would handle the following page access pattern: $A, B, C, D, E, A, B, E, D, B, B, A$. As in the figures below, assume that the system has three frames of memory (each which can hold a single virtual page). Fill in the frame contents for each step of the access pattern and report the total number of page faults for each algorithm.*

FIFO	A	B	C	D	E	A	B	E	D	B	B	A
F1												
F2												
F3												
Fault?												

MIN	A	B	C	D	E	A	B	E	D	B	B	A
F1												
F2												
F3												
Fault?												

Question 5. *(25 points) Consider the file architecture shown in the following figure. For each file there is an index structure which contains 8 entries: the first 7 entries are pointers to 1KB data blocks and the last entry points to a secondary index data structure which contains 8 more data pointers.*



a. *(5 points) What is the maximum size of a file using this architecture?*

b. *(10 points) If you increased the number of entries in both data structures from 8 to 32 (but still kept only one index pointer in the Index), how would that impact the size of allowable files? Would it impact the performance of file accesses (and if so, how)?*

c. *(10 points) What simple change could you make to this structure to support files of infinite size? Is there any downside to this (modified) approach?*

Question 6. *(10 points) Using the following set of disk requests, what is the order of seeks when using the SSTF and SCAN disk scheduling algorithms? What is the total distance of seeks in each case? Assume that the disk has its head at position 45 out of 100 and that it is currently moving towards higher numbers (for SCAN).*

Disk Request Queue: *44, 57, 78, 65, 46, 90*

Question 7. *(10 points) Suppose you are designing a filesystem to run on SSDs. Since there is no longer any rotational delay, is there any downside to scattering blocks of individual files all over the disk, as opposed to storing them in a consecutive series of sectors? Why or why not?*