1. (+12) Describe at least 3 general approaches in memory management  
than can help solve the external fragmentation problem.

One solution for external fragmentation is to use variably sized segments, segmentation.

A complimentary solution to segmentation is to use paging which allows the physical address space of a process to be non-contiguous.

Another solution is compaction where we try to move all free space to one end of the memory.  
  
2. (+12)  
a. 1603 bytes using a best-fit policy? **2200 bytes**  
b. 949 bytes using a best-fit policy? **1000 bytes**  
c. 1603 bytes using a worst-fit policy? **2200 bytes**  
d. 349 bytes using a worst-fit policy? **2200 bytes**  
e. 1603 bytes using a first-fit policy? **2200 bytes**  
f. 1049 using a first-fit policy? **2200 bytes**  
  
3. (+20)

P1 P2

|  |  |
| --- | --- |
| page | Frame |
| 0 | X |
| 1 | Y |
| 2 | A |
| 3 | B |
| 4 | C |
| 5 | D |
| 6 | E |

|  |  |
| --- | --- |
| page | Frame |
| 0 | X |
| 1 | Y |
| 2 | P |
| 3 | Q |

Paging does not need to be contiguous frame X and Y can be in the same frame and all others have unique frames in memory.

4. (+12)

TLB hit = T \* P\_TLB -> 0.9ns. TLB miss & page hit = (1-P\_TLB)(1-p)M -> 0.999ns. TLB miss & page fault = (1-P\_TLB)(p)D -> 1000ns. 0.9+0.999+1000 = 1001.899ns

5. (+12) The Least Recently Used (LRU) page replacement policy does not  
suffer from Belady's Anomaly. Explain intuitively why this is the  
case. Construct an example page fault sequence to illustrate your  
point.

Because the LRU policy marks when pages have been accessed recently and does not replace the pages recently used, but replaces the page that was accessed the longest time ago. Hopefully this page is not needed vs the other pages in memory because it was the last accessed of all pages.

Instead of replacing the first page in like FIFO the used page is marked and will not be flushed.

6. (+20) Given a frame allocation of 3, and the following sequence of  
page references 3 2 4 3 4 2 2 3 4 5 6 7 7 6 5 4 5 6 7 2 1, and  
assuming main memory is initially unloaded, show the page faulting  
behavior using the following page replacement policies. How many   
page faults are generated by each page replacement algorithm? Which  
generates the fewest page faults?

a. FIFO 12 including the initial 3

324-456-567-674-745-456-567-672-721

b. OPT 10 including the initial 3

324-354-654-657-654-675-672-721

c. LRU 10 including initial 3

324-524-564-567-564-567-672-721

7. (+12) Assume the same sequence of page references as in problem #6,  
and assume memory is initially unloaded, but now assume that a dynamic  
paging working-set algorithm is applied to the same sequence of  
page references, with a window size of 6. Draw the page faulting  
behavior. Your solution chart should show the frame allocation at  
any given time to the process.

3 2 4 3 4 2 2 3 4 5 6 7 7 6 5 4 5 6 7 2 1

{324} set until position 9 and 5 is added -> {3245}, then 6 is added -> {32456}, then 7 is added -> {324567} and now we have a full window of 6 pages. We then reach the 1 at the end of the sequence and remove 3 because it is the oldest and least used from the set -> {245671}.