Extra Credit 2 Answer

Due: May 9, 2022 **Points:** 20

Remember, you must *justify all your answers*.

1. (*20 points*) Prove that Belady's anomaly cannot occur in a stack algorithm.

   Answer: Let n be the number of frames and $M(n)$ the set of their contents.

   The key here is to characterize Belady's anomaly generally; it is simply that as the number of page frames increases, page faulting increases. This means that the page being brought in was not in memory before it is brought in as a result of the memory size allocation changing to $n+1$. If all pages in the memory $M(n)$ were present in the memory of $n+1$ frames $M(n+1)$, this could not happen; the number of page faults would either be constant or decrease.

   To see this, consider what happens when a page fault occurs in a stack algorithm. One of 3 cases arise:

   (a) The page fault occurs in the $M(n)$ memory but not in the $M(n+1)$ memory. This means that the page coming into the M(n) memory is in the $M(n+1)$ memory, as before the page fault, every page in $M(n)$ is in $M(n+1)$) by definition of stack algorithm. So there is one more page fault in the $M(n)$ memory than in the $M(n+1)$ memory.

   (b) The page fault occurred in the $M(n+1)$ memory but not in the $M(n)$ memory. This means the process needed a page not in the $M(n+1)$ memory but that was present in the $M(n)$ memory. This means that $M(n) \nsubseteq M(n+1)$ because if the page was in $M(n)$, by definition of stack algorithm, it must be present in $M(n+1)$. This contradicts the assumption that we are dealing with a stack algorithm, and hence cannot happen.

   (c) The page fault occurs in both $M(n)$ and $M(n+1)$. Then one page fault occurs in both.

   So, for a stack algorithm, the number of page faults in a memory $M(n)$ is equal to or greater than the number of page faults in $M(n+1)$.But Belady's anomaly is exactly the opposite, that is, the number of page faults in $M(n)$ is greater than the number of page faullts in $M(n+1)$,

   Hence Belady's anomaly cannot occur in a stack algorithm.