Section 7
November 9, 2017

Swapping Pages to Disk:
In this lab you will be focusing on implementing an algorithm to swap pages to memory when the kernel runs out of physical memory. We talked extensively about paging algorithms in class, your goal will be to implement an approximation of LRU for xk. When we say best approximation, we are looking for behavior that doesn't evict pages that are recently use. Since the tests do not inspect the kernel, they can only look at the behavior and infer what is happening.

Designing:
There is almost no scaffolding for this lab, we highly encourage you to create a thorough design *before* you start coding. There are many edge cases that you can run into, and trying to flesh them out on paper will help you have a better understanding of your model.

Questions (from the lab specification):
* Use diagrams if necessary
* You might need to define extra data structures in xk to realize those functionalities. You might also need to use extra bits in the page table entry. You can safely use bit 10 and bit 9 in the page table entry, as those are not used by hardware.

When should we flush pages to swap region and when should we load them back?
How should we keep track of a memory page that is in swap region?

Section 7
November 9, 2017

What should happen when a swapped memory page is shared via copy-on-write fork?
Is there a set of memory pages you don't want to flush to swap?
What will happen when forking a process some of whose memory is in the swap region?
How will the page table entry change for memory pages that are swapped out?
What will happen when exiting a process whose memory is in the swap region?