Mentor notes

- Question 1 is basically to step mentees through the process of comming up with the clock algorithm (or something similar to it). Many mentees might come up with a solution similar to an LRU. Walk through them why this would be a bad idea for pages. Basically, it would require timestamping every access or write to a page, which would get really expensive (memory is being modified all the time).
- In Q1, talk about the “dirty” bit a bit for a bit (heh). Basically, evicting pages that are clean and have been in swap before can be really cheap, since you won’t have to write it out to disk. It’s a nice optimization.
- For question 2, I haven’t gotten a chance to graph out the two programs. However, once you get to around 100,000 in my implementation (which uses a C++ list and vector that start out with size 0 and I just use `push_back` to add elements), the difference in both runtime and cache hits becomes significant. In my system at 100K elements, the linked list implementation took ~0.017s vs ~0.0071s for the vector implementation, and a 73.11% L1 cache miss rate vs 0.61%. Pointing this out, and mentioning that they’re both still O(n) could prove extremely useful.