PIC 10B Discussion
Week 8
Professor: Michael Lindstrom
TA: Thomas Tu
Data Structures: Linked Lists

• Exercise: reverse a linked list
  • 1,2,3,4,5 -> 5,4,3,2,1

• Remember to handle both first and last pointers

• Also handle edge cases
  • What if the list is empty? What if it has only one element?

• std::swap may come in handy
Iterators

• From the outside, LinkedList will not let you access or change individual nodes and their attributes
• Interface: iterator (intuitively, a fancy index)
• Allows you to do for each loop
  • iterator insert(iterator, int)
    • Returns iterator to inserted element
  • iterator erase(iterator)
    • Returns iterator one after erased element
Exercise

• Easy exercise: search a linked list using iterators
  • Range based for loop

• Actual exercise: Reverse a linked list using iterators
  • Helpful: std::iterswap(iter1, iter2)
    • Equivalent to std::swap(*iter1, *iter2)
  • Think of a different algorithm than you used with nodes
Why Iterators?

• Generalized container index
• Can create functions that handle all kinds of containers, no matter how they’re implemented
• For example, std::search, std::reverse
• Philosophy: it shouldn’t matter how you did it, but what I can do with it; so long as it does the right thing and doesn’t break, I don’t care
  • Give me an interface, outline all the edge cases, and the code does that
• Alternate philosophy: I do care how it’s done, because of efficiency
  • Node swapping faster than value swapping for larger data type values