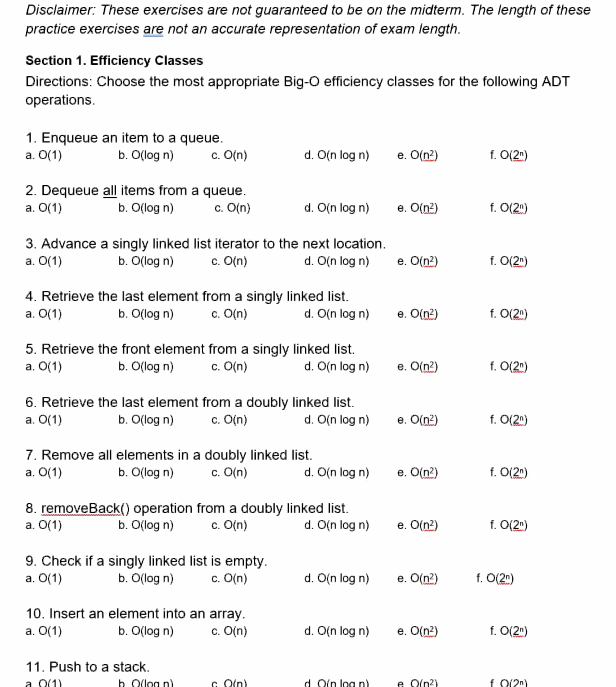
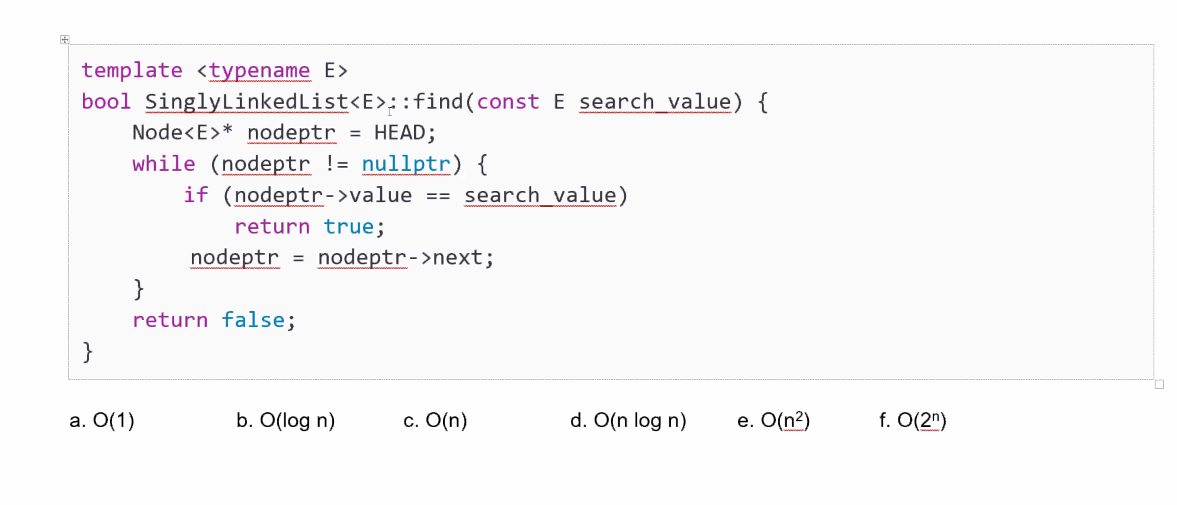
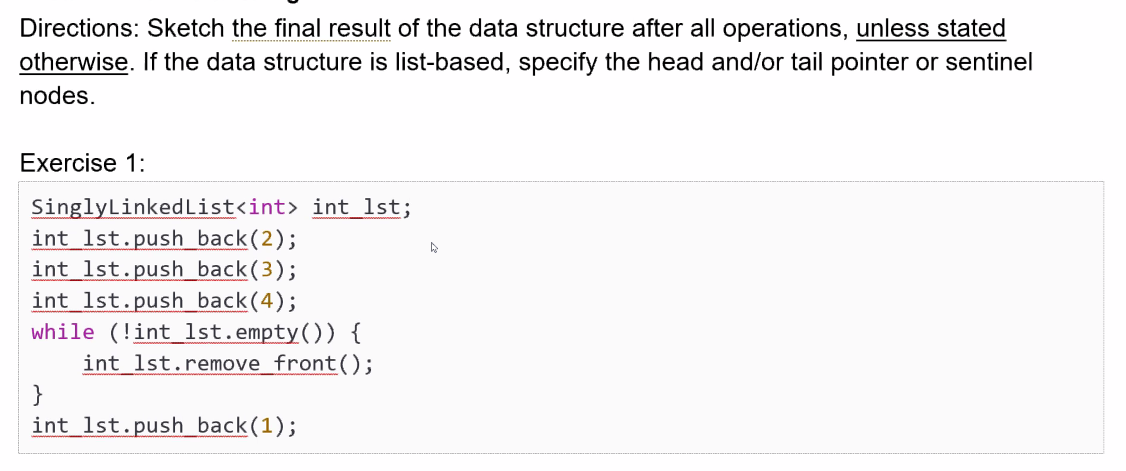
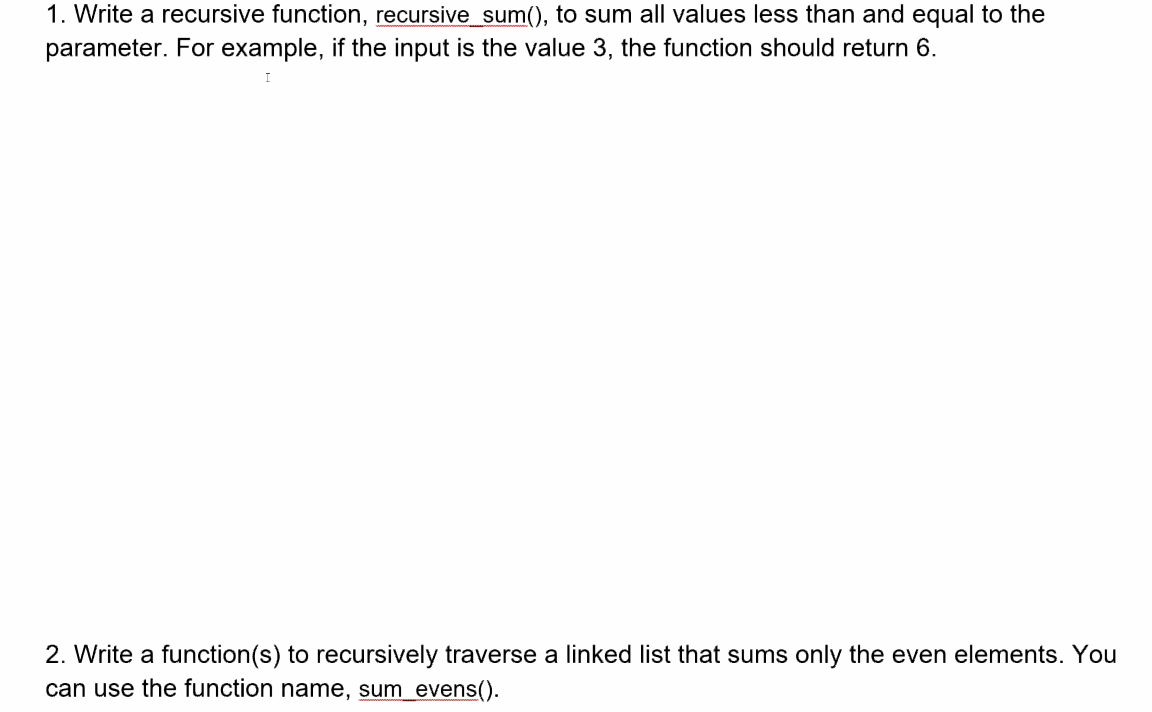
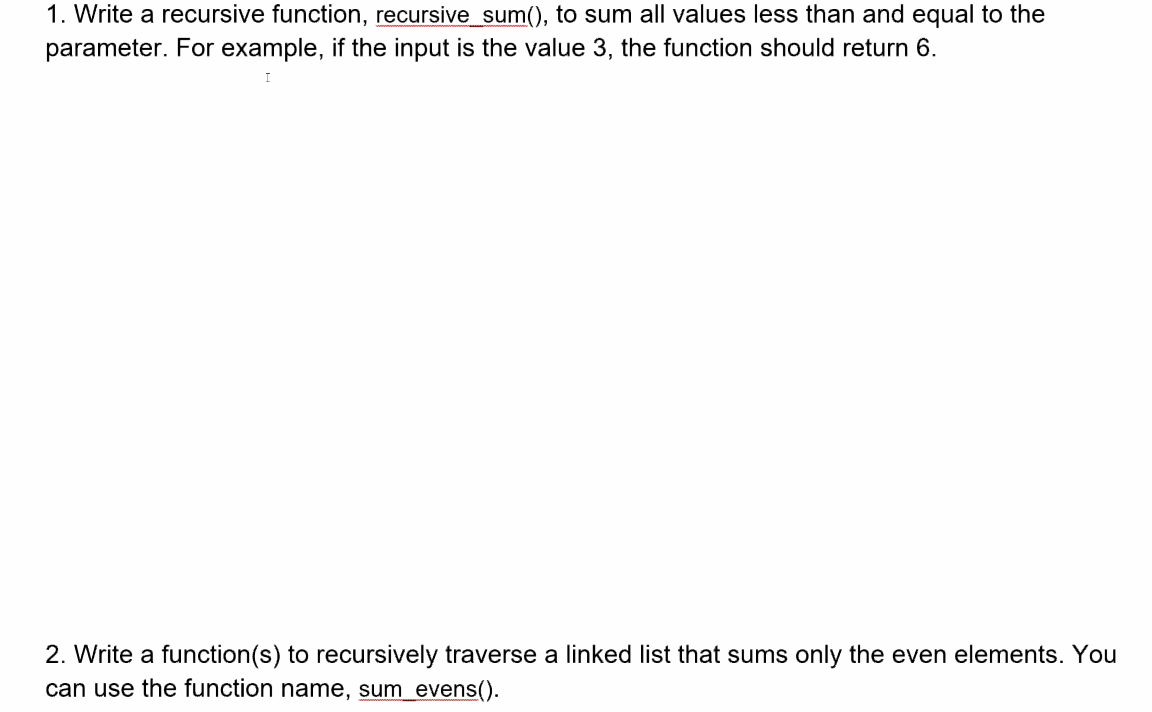
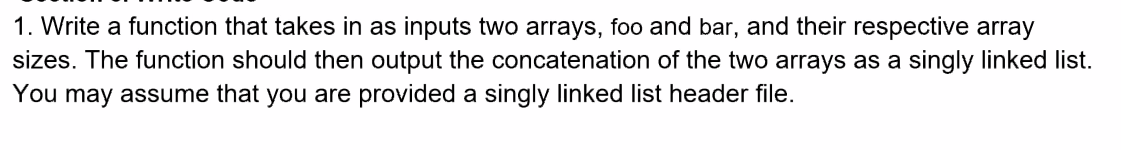
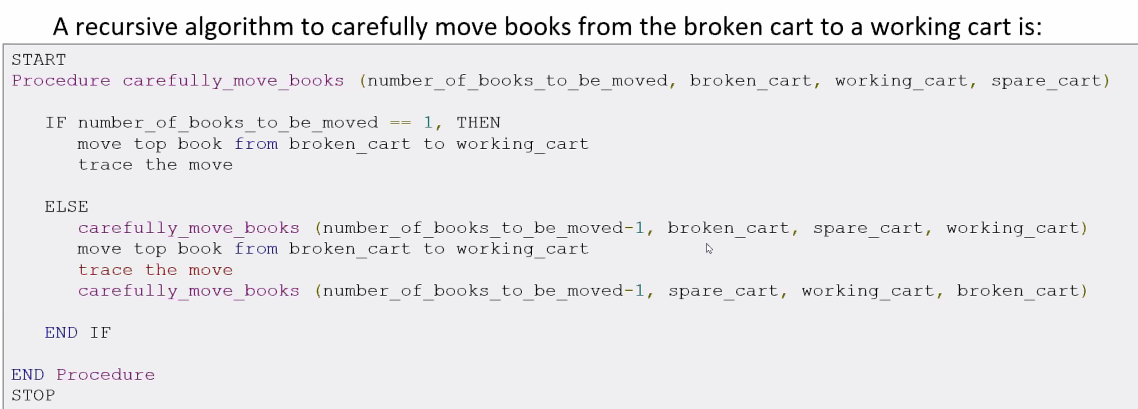
Lecture 14

CPSC 131  
10/14/2020

1. Lots of things due next Monday
   1. Knowledge check 3
   2. Project 2/3
2. Midterm Logistics
   1. Must have working camera. KEEP ZOOM ON!
      1. No Proctorio
   2. Hold up ID to camera for at least 5 seconds
   3. Format
      1. Closed books. Closed notes, no internet surfing, just like checkpoints
      2. Four parts to the exam
         1. Multiple choice, true/false, short answer
         2. Data structure sketching
            1. Sketch by hand or Photoshop but you need to save it as a JPG and load it into the Canvas
         3. Analyzing alternatives essay
            1. Choose your favorite text editing software to essentially write an essay with code and don’t forget the Big-O notation/Analysis!
      3. Coding
         1. Add and implement a new member function to one of our existing abstract data type classes taken from our **Implementation Examples**
         2. Your implementation may require you to add an overloaded helper function as well
         3. Your code must be syntactically and semantically correct
3. Midterm Topics Covered
   1. Part 0 - Introduction & Review
      1. C++ pointers, references, arrays, dynamic memory, Object Oriented Programming (OOP), classes/structs, templates, exceptions
         1. Allocating things on the stack vs heap
      2. Writing/updating classes with proper
         1. encapsulation (public/private
         2. instance attribute data members
         3. constructors, destructors
         4. overloaded queries, accessors, and mutators
         5. overloaded operators
      3. Algorithm complexity analysis
         1. asymptotic analysis (big-O)
         2. efficiency classes for the fundamental operations of all the data structures covered  
             •O(1), O(log2 n), O(n), and O(n2)
         3. choosing a container by comparing efficiency classes of operations
      4. Memory model (stack vs. heap)
      5. Iterative and recursive algorithms
   2. Part 1 - Sequence Containers
      1. Arrays & Vectors
         1. Fixed, bounded, & unbounded implementations
         2. Amortized efficiency, complexity analysis
      2. Lists
         1. Singly & doubly linked lists
         2. Null-terminated, two dummy nodes, circular with one dummy node
         3. Complexity analysis
      3. Concepts & Interfaces
   3. Part 2 – Iterators
      1. Iterator Concepts & Interfaces
      2. Pointers as iterators
      3. Container Traversal Techniques
         1. Iterative & recursive
   4. Part 3 - Container Adapters
      1. Stacks, Queues
      2. Concepts & Interfaces
      3. Array, Vector, List implementations
         1. Fixed sized arrays
         2. Bounded and unbounded vectors
         3. Singly and double linked lists
      4. Complexity analysis
         1. Fixed sized arrays
         2. Bounded and unbounded vectors
         3. Singly and double linked lists
      5. Complexity analysis
4. Midterm Sample Questions
   1. Part 1: Sample Questions   
      
      1. 4. C. O(n)
      2. 10. C. O(n)
      3. 11. A. O(1)
      4. 14. A. O(1)
   2. Part 2: Efficiency Analysis Sample  
        
        
      
      1. 5. C.O(n) [the final unnumbered one]
   3. Part 3: ADT Sketching  
        
        
      
      1. Exercise 1  
         Part 1: push\_back 2 3 4   
         Part 2: remove\_front  
         Part 3: push back 1
      2. Exercise enqueue
   4. Part 4: Recursive  
        
      
   5. Part 5: Write Code  
      
   6. Part 6  
      It’s more code to write down
5. Project Book Cart
   1. Recursion
      1. We have to move our books from Cart A to Cart B while using Cart C as a transitional stage. It is a recursive algorithm  
         
      2. This changes the order of parameters
      3. Heavy book should never be on top of the light book
      4. Eventually all will be in working cart in the right order.
   2. Stack and Queues
      1. We scan book by **ISBN**
      2. If the book isn’t found in database, it’s free
      3. The Database aspect is implemented first
      4. Constructor needs to open text file containing Books and populate a memory resident data store with contents of text file
      5. Read as BOOKS
      6. Populate data base, implement find() to look for book recursively
      7. Color in the lines. DON’T WRITE OUTSIDE OF THE //To-Do// //End // THINGS!
   3. The Code itself
      1. BookDatabase.cpp
         1. Populate the vector with what you read
         2. Go back to P0 and see how we read it in.
         3. Not cin but fin
         4. ToDO3 is Implementing the functions find() and size()
      2. BookDatabase.hpp
         1. Yes there are TODOs in hpp
      3. Main.cpp
         1. Got to use the trace() functions