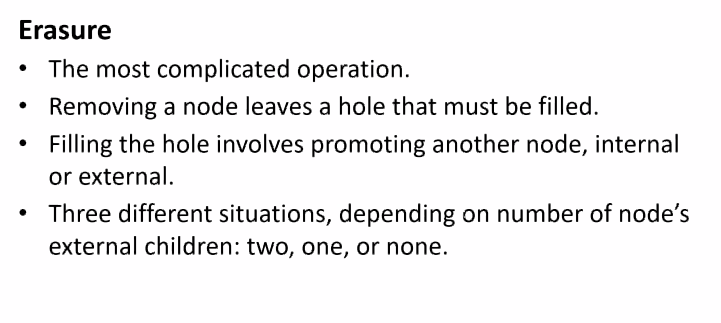
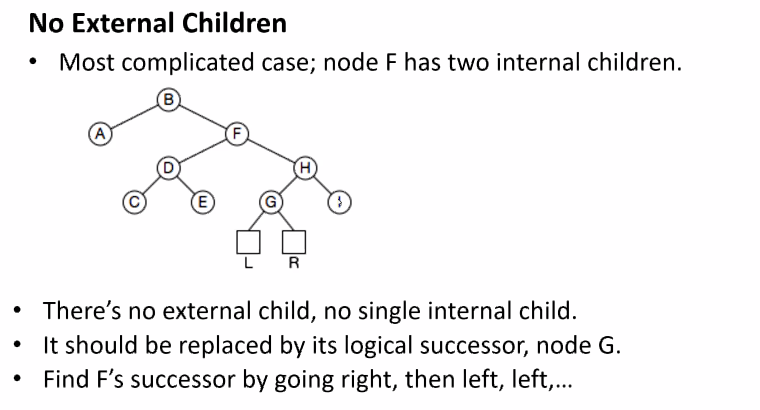
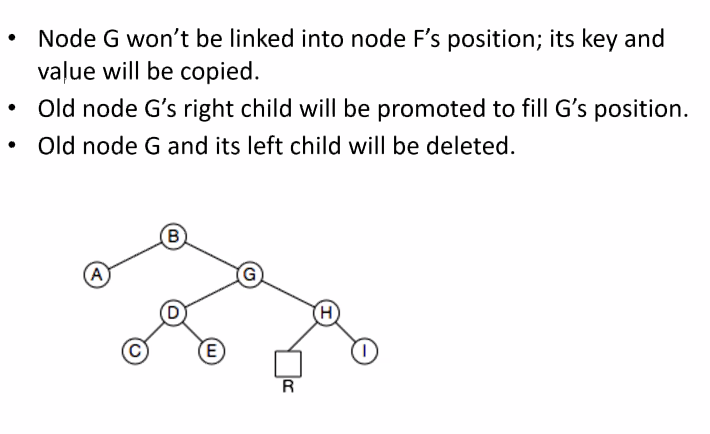
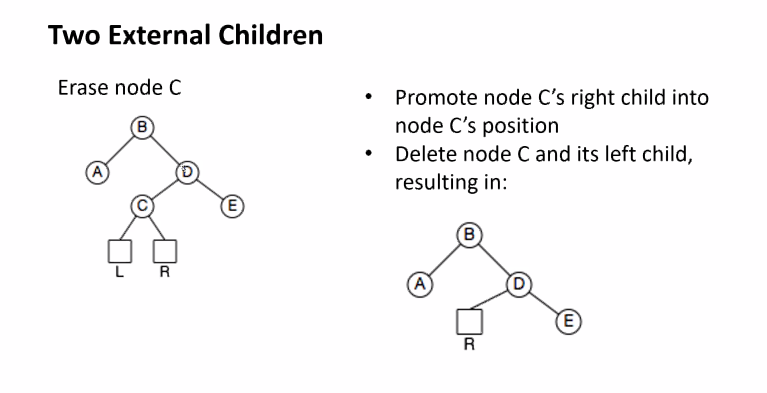
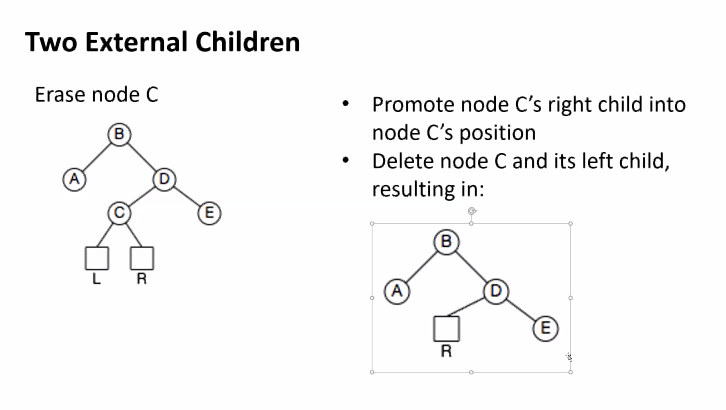
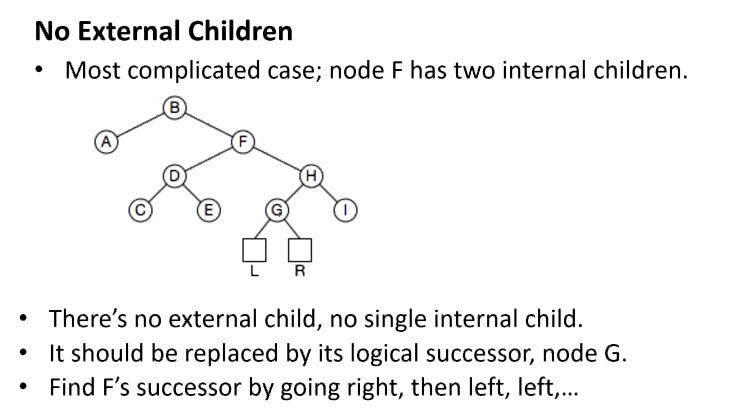
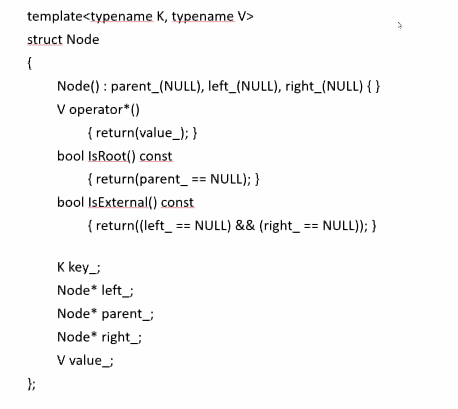
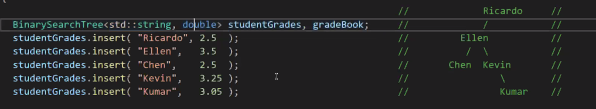
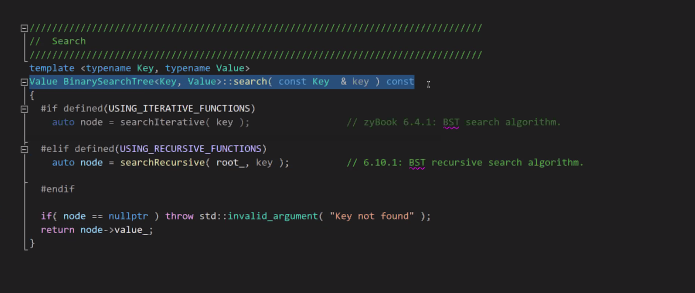
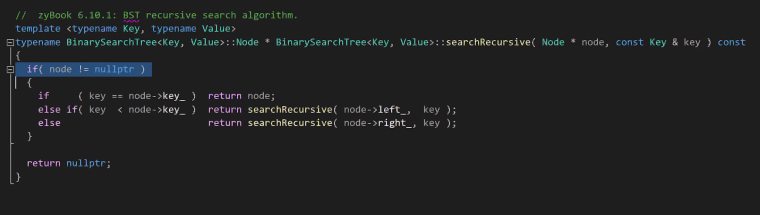
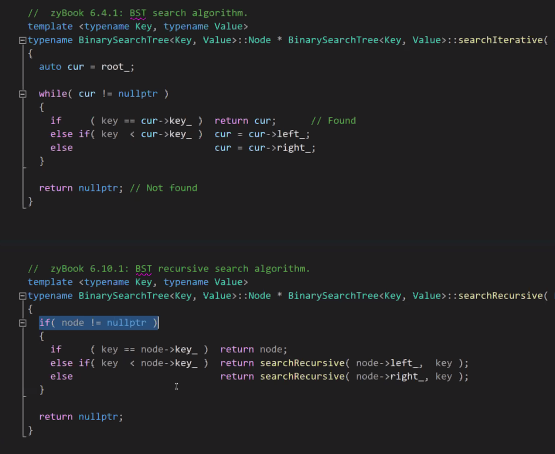
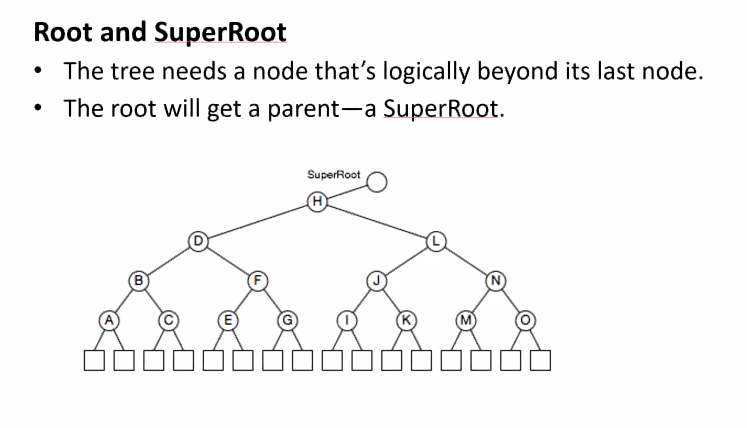
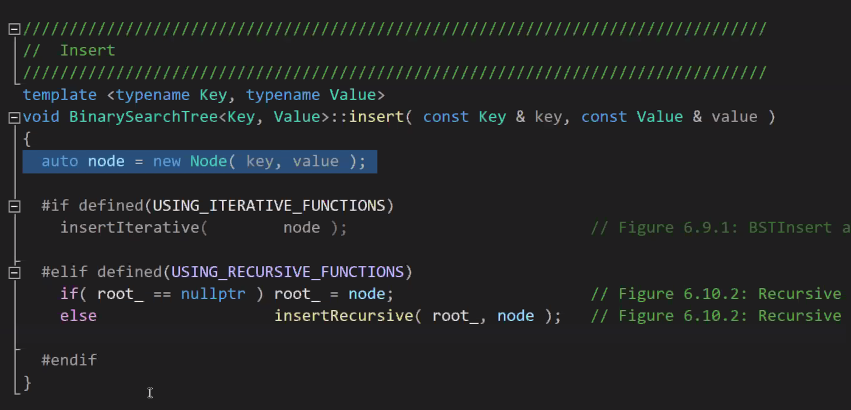
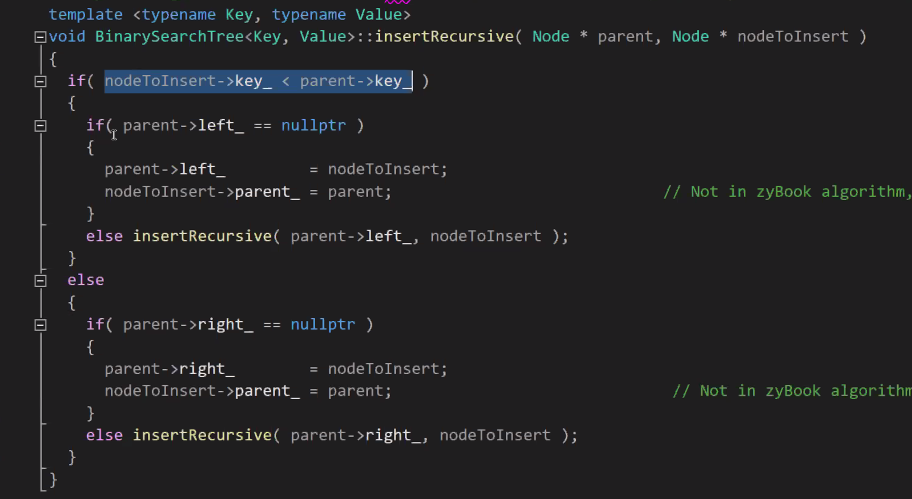
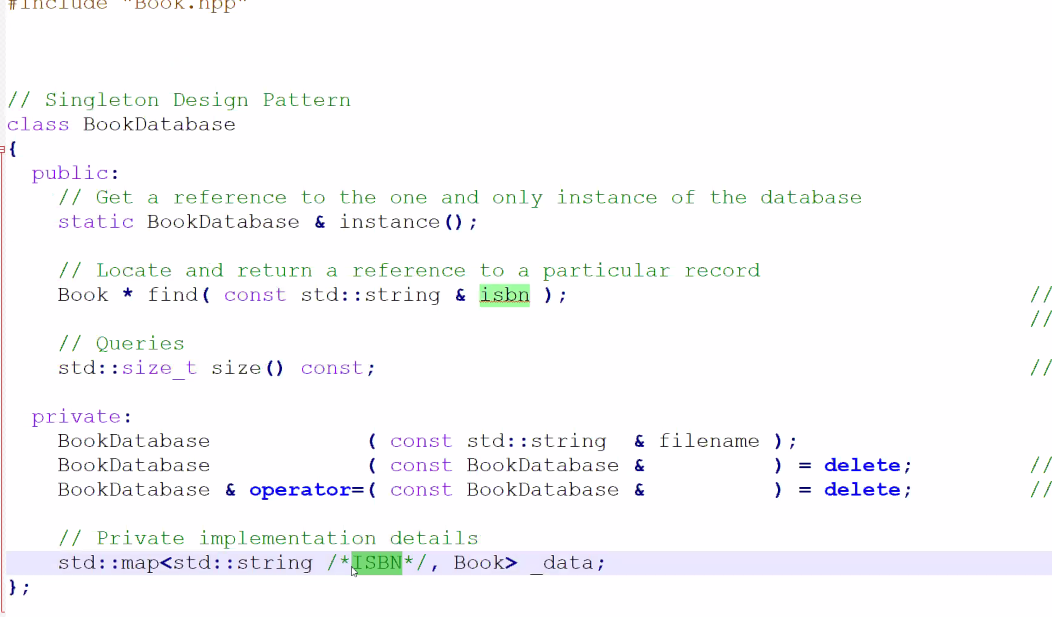
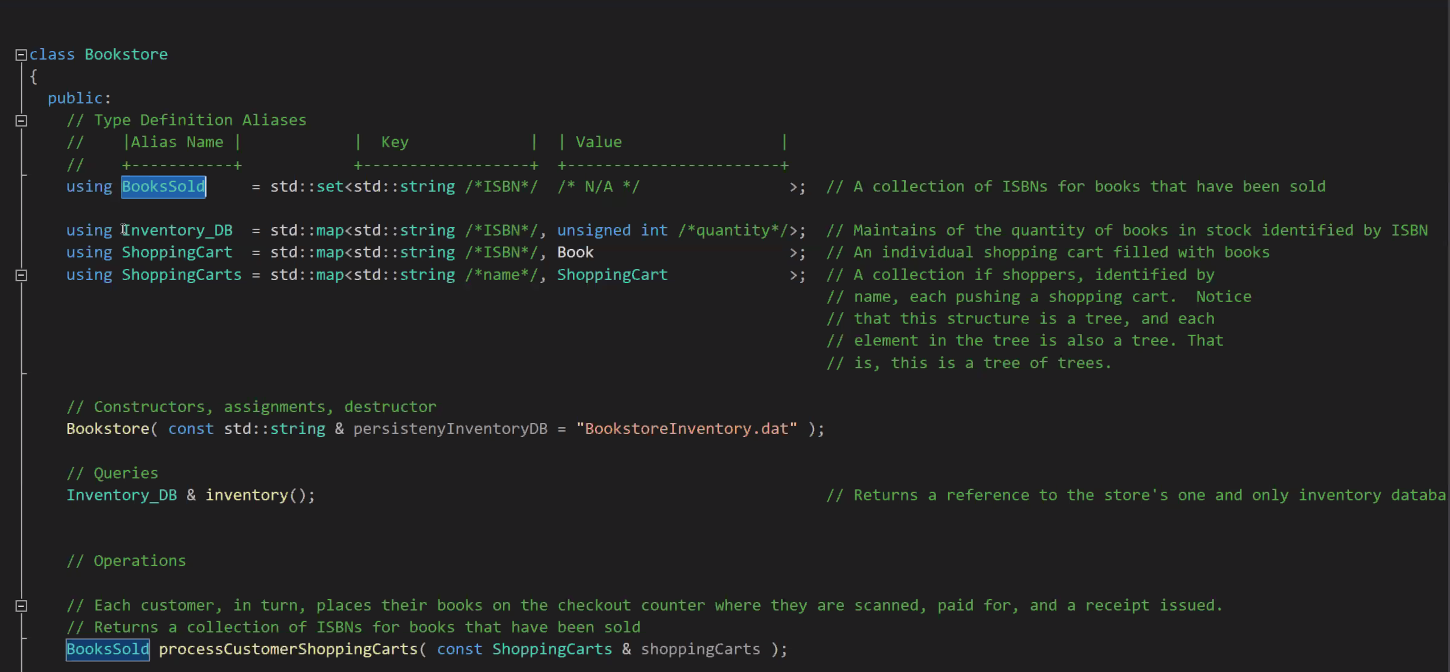
Lecture 18

CPSC 131  
11/4/2020

1. Trees
   1. Erasure  
      
   2. No external Children  
      
   3. NOOOODES  
      
      1. By moving the successor up, what we’ve accomplished is that everything on the right is still attacked. G was less than H because it was to the left of H. So if we move it to the parent of H, the less than property still holds.
      2. So to delete the element from the binary tree, we have to do it in steps.
      3. If the node you’re trying to delete has 0,1,2 children, it makes a difference on how the node is removed.
   4. If we remove a node with 0 children,   
      
      1. This is easy
   5. IF we remove a node with 1 child  
      
      1. Remove D, relink E to B
   6. IF we delete node with 2 kids  
      
      1. Delete successor, make node on right the new parent to L
   7. Code   
      
   8. Tree  
      
   9. The Search (public) function calls its private recursive version   
      
      1. This is the most important function as the rest of the other functions depend on Search
      2. How we kick off this function, we go to the top of the tree to start.
      3. Biggest question: Did I find what I’m working for?
      4. If no, throw under exception as we can’t return a default value.
      5. If yes, see below
      6. Binary Search Tree Recursion  
         
      7. Base Case + End case  
         
      8. The General code  
         
   10. Insert
       1. It’s okay to use auto  
          
       2. The Rescursive insert  
          
2. Project
   1. The project has a couple of things: BookStoreDatabase will want to change from the vector to a MAP   
      
      1. Associative Container: give me a key, return a value. Key = ISBN. Value = Book
      2. Imeplement BinarySearchTree to the FIND function to get the key and the value of the book
      3. Don’t write code to traverse a tree. Demonstrate how to use a STD:MAP function to find a book.
      4. Implementation (TODO3): Implement Find function
         1. Returns a pointer to book
         2. Trying to implement as a function that searchs the MAP we created
         3. Don’t walk the list from beginning to end.
         4. It’s no longer a recursive function.
         5. Use find from library and convert results from pointer to book
   2. BookStore
      1. 
      2. Using Inventory\_DB will help document meaning with code