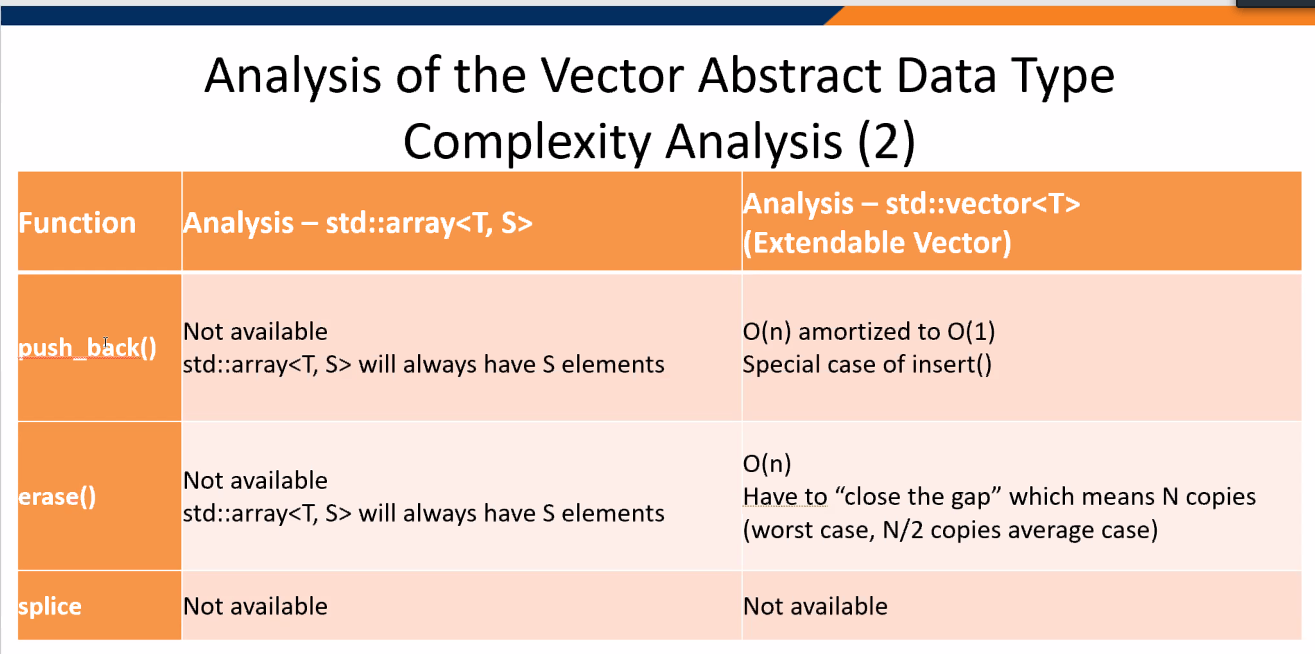
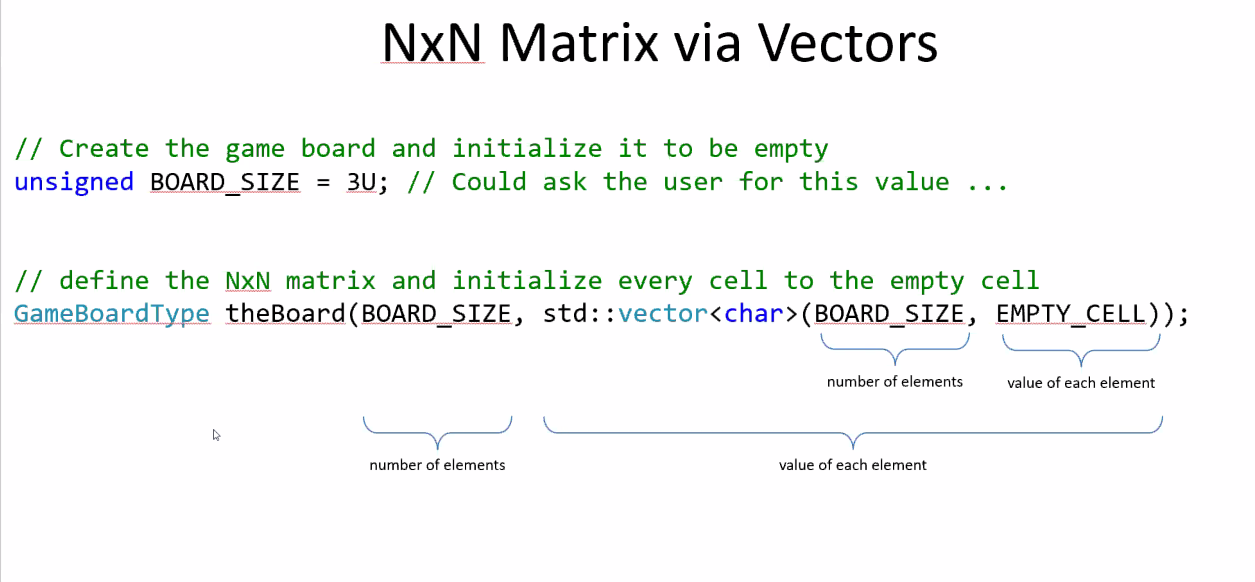
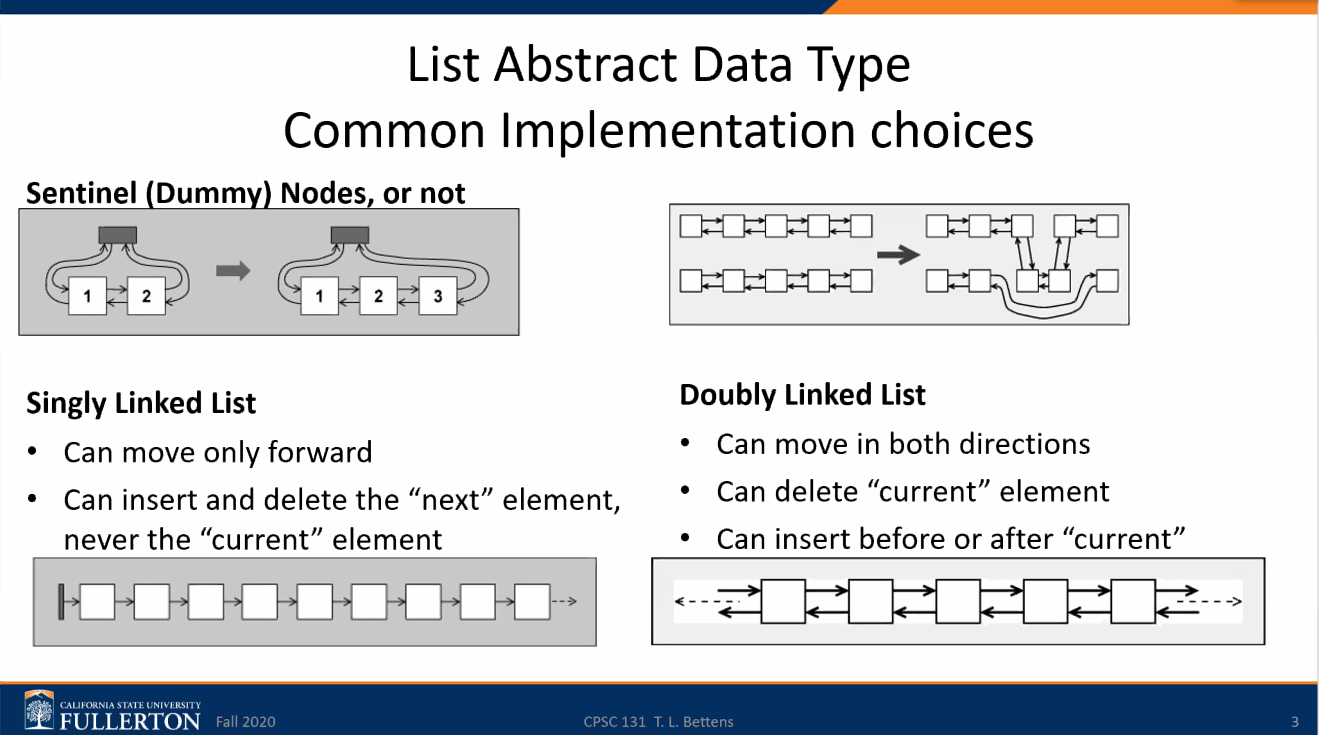
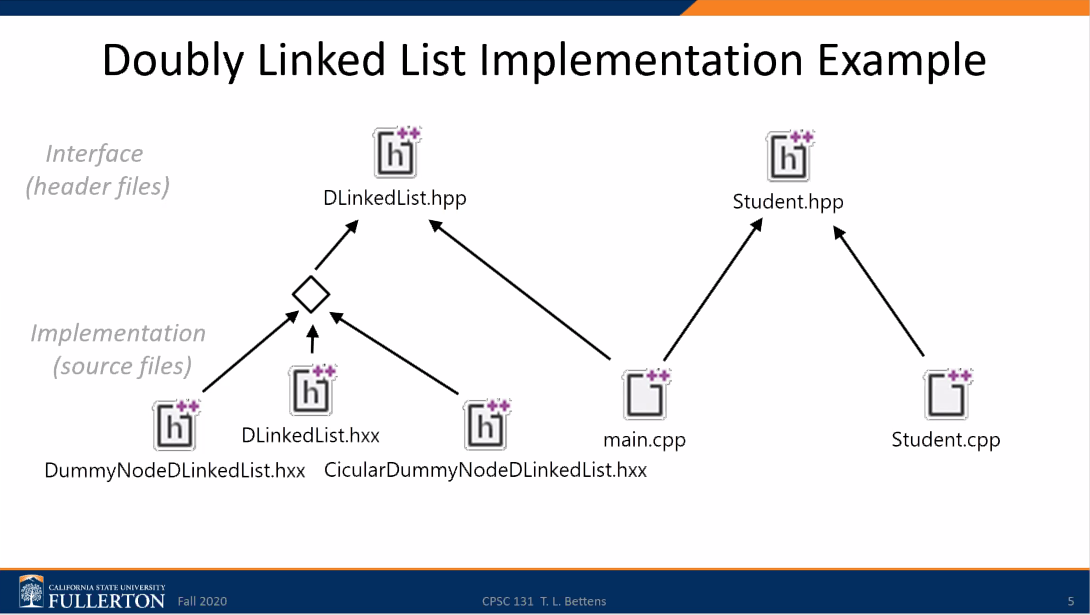
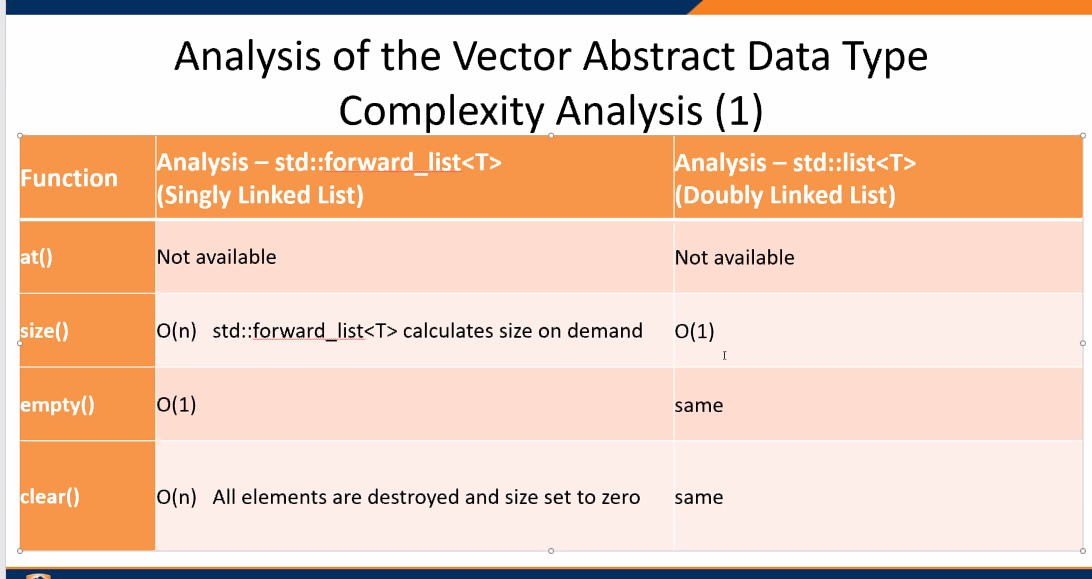
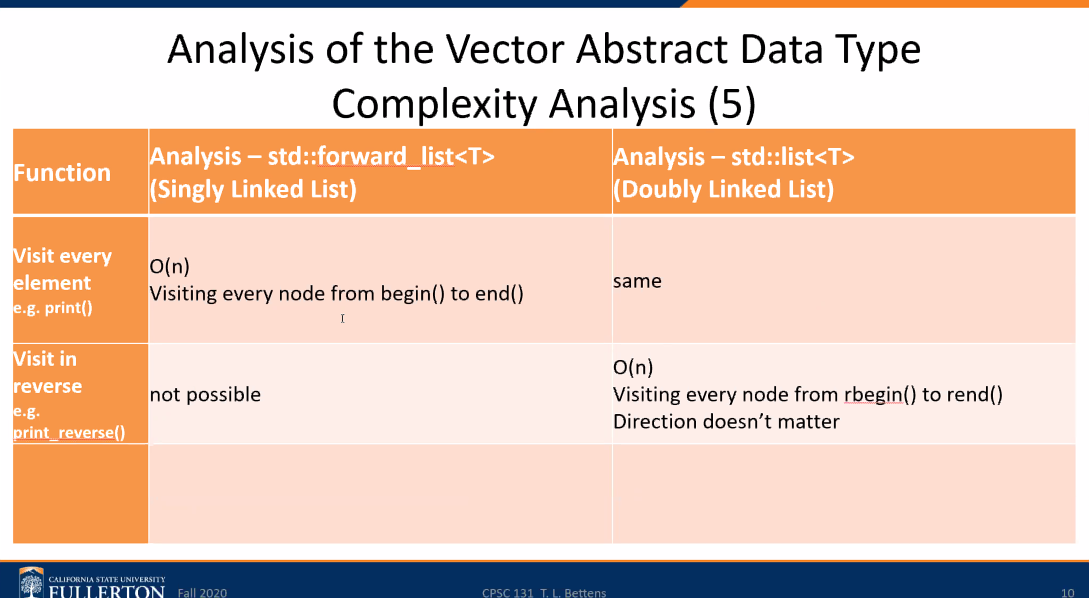
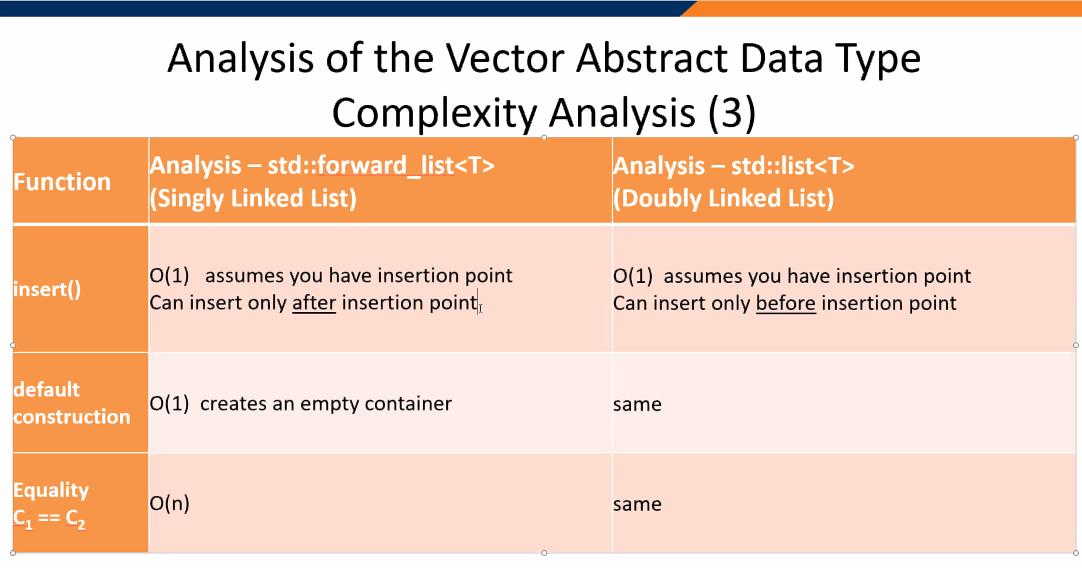
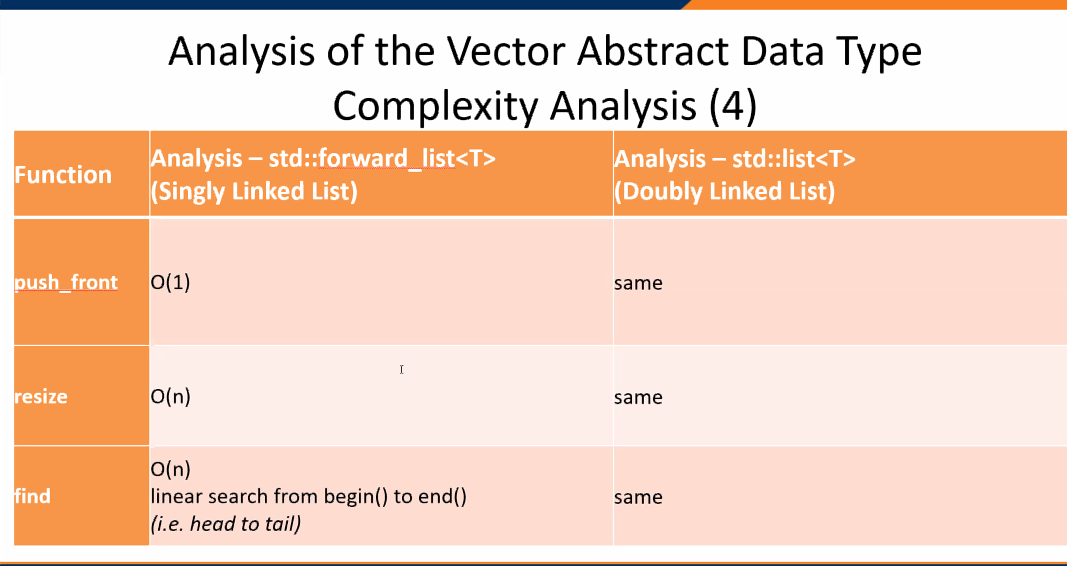
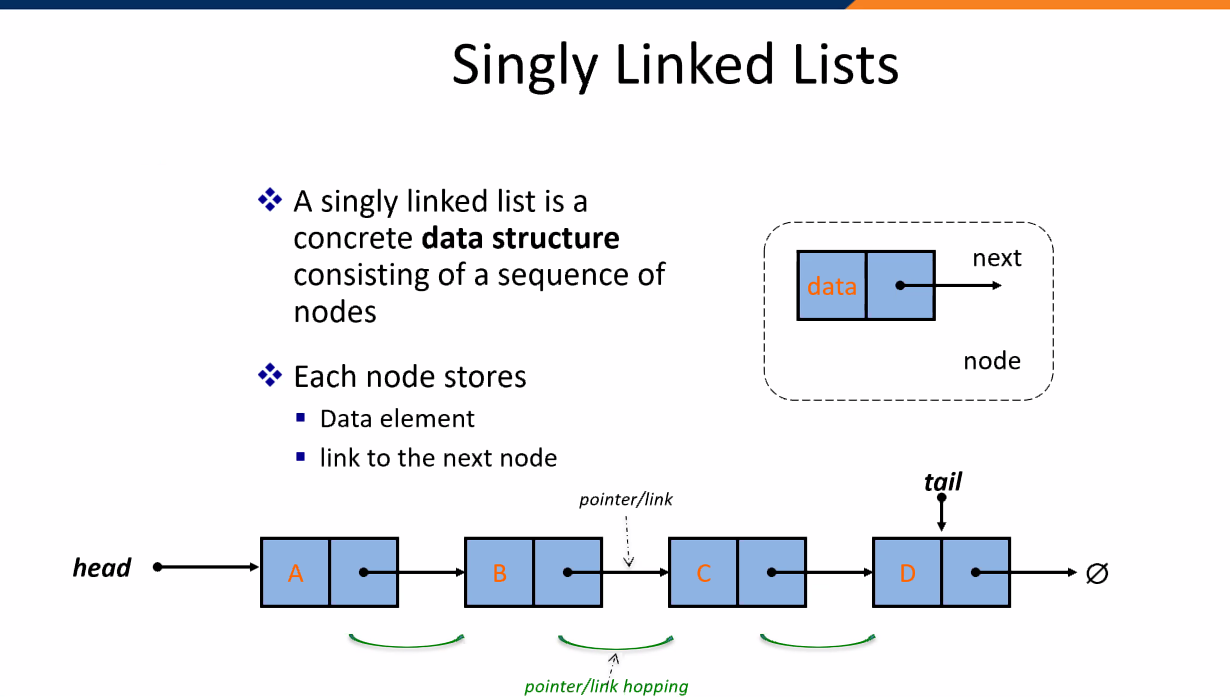
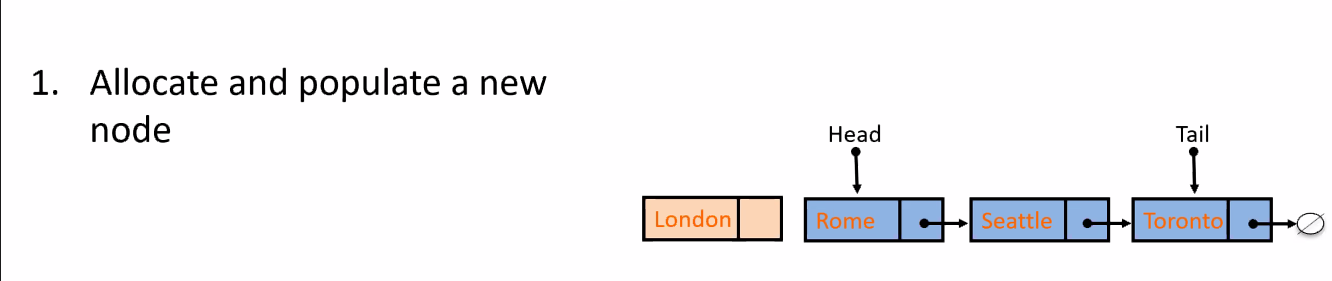
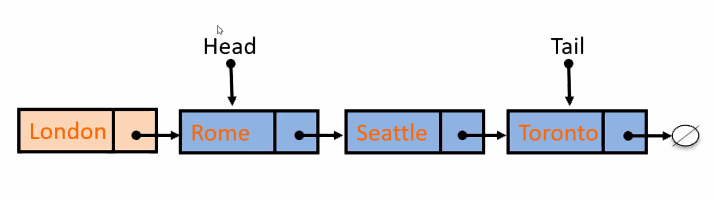
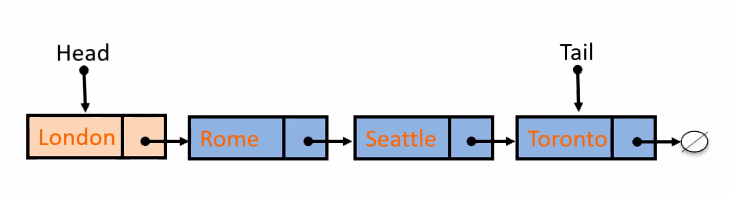
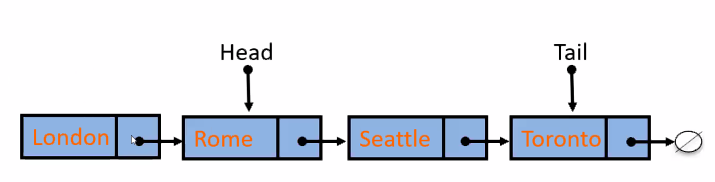
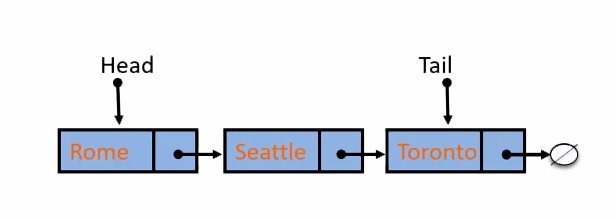
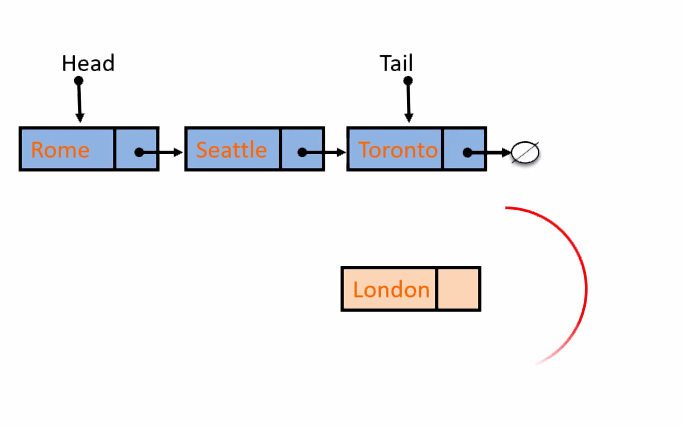
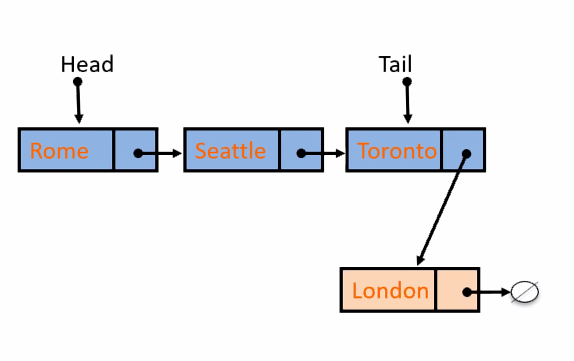
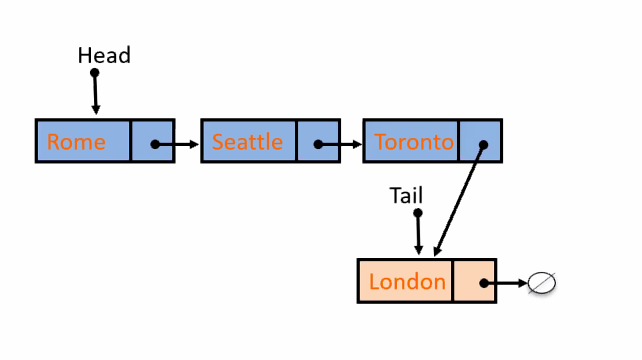
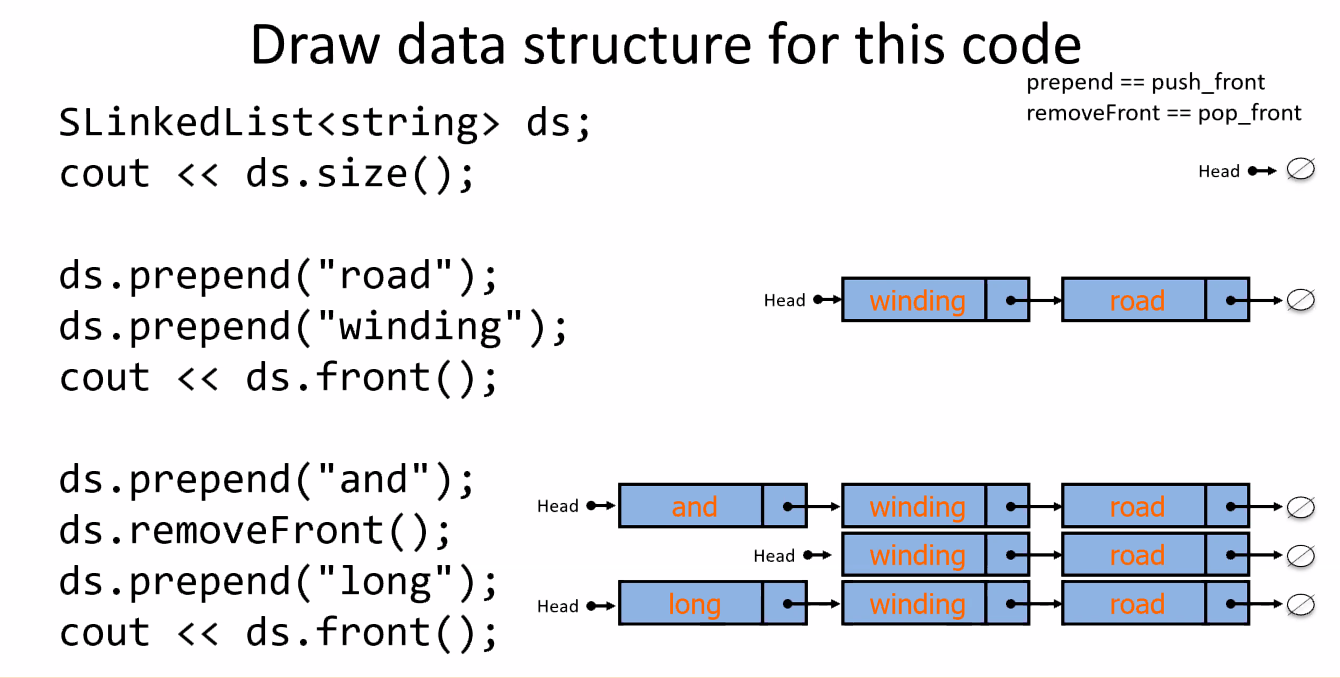
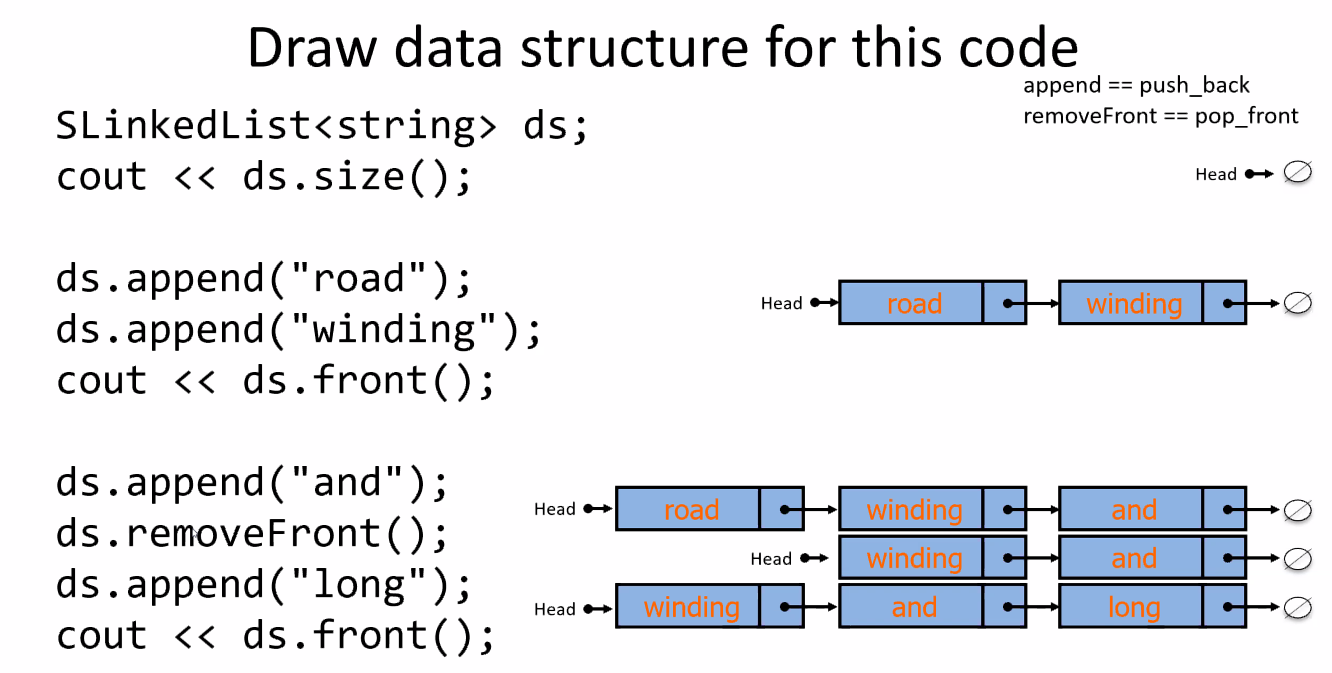
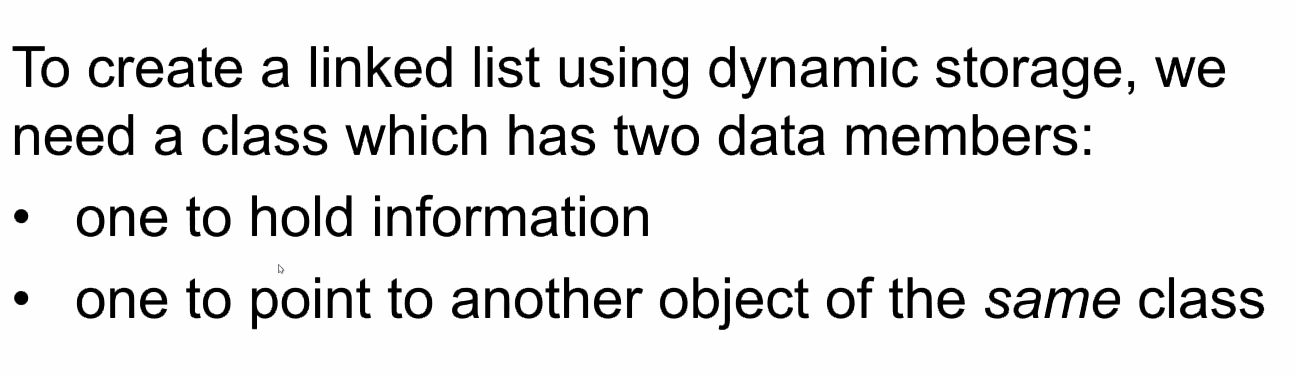
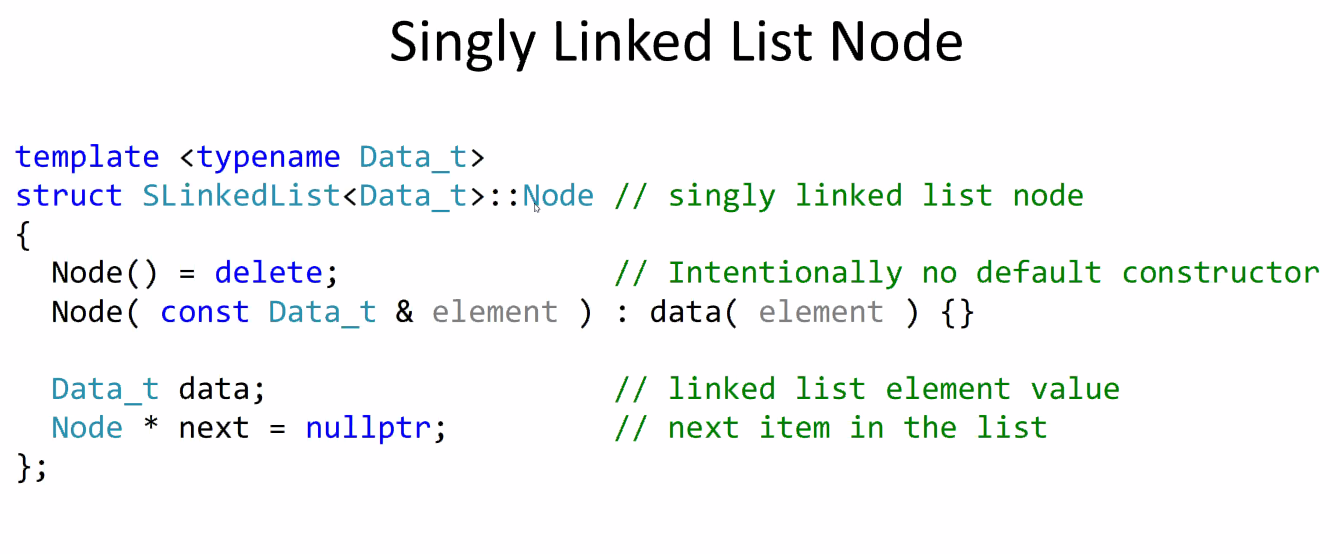
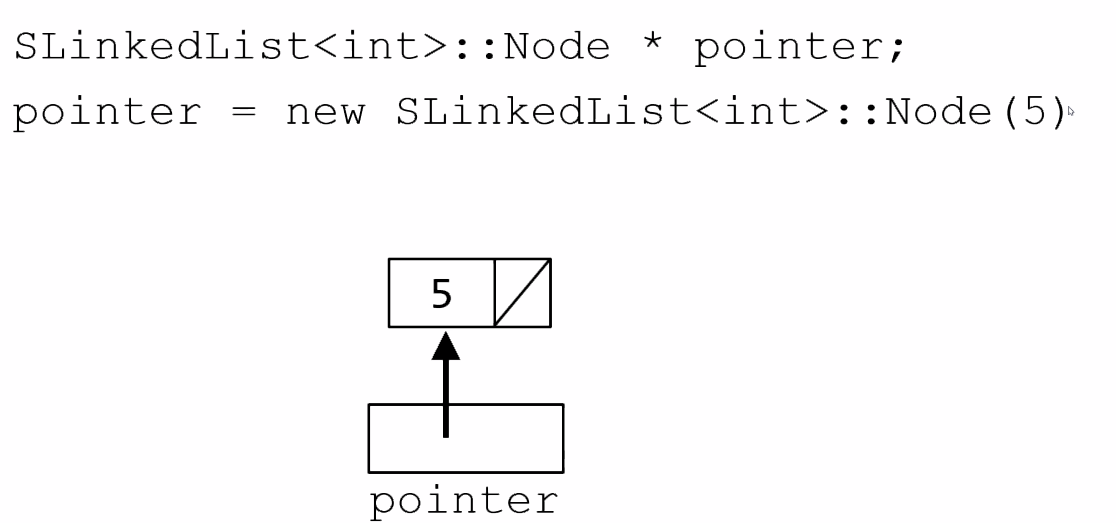
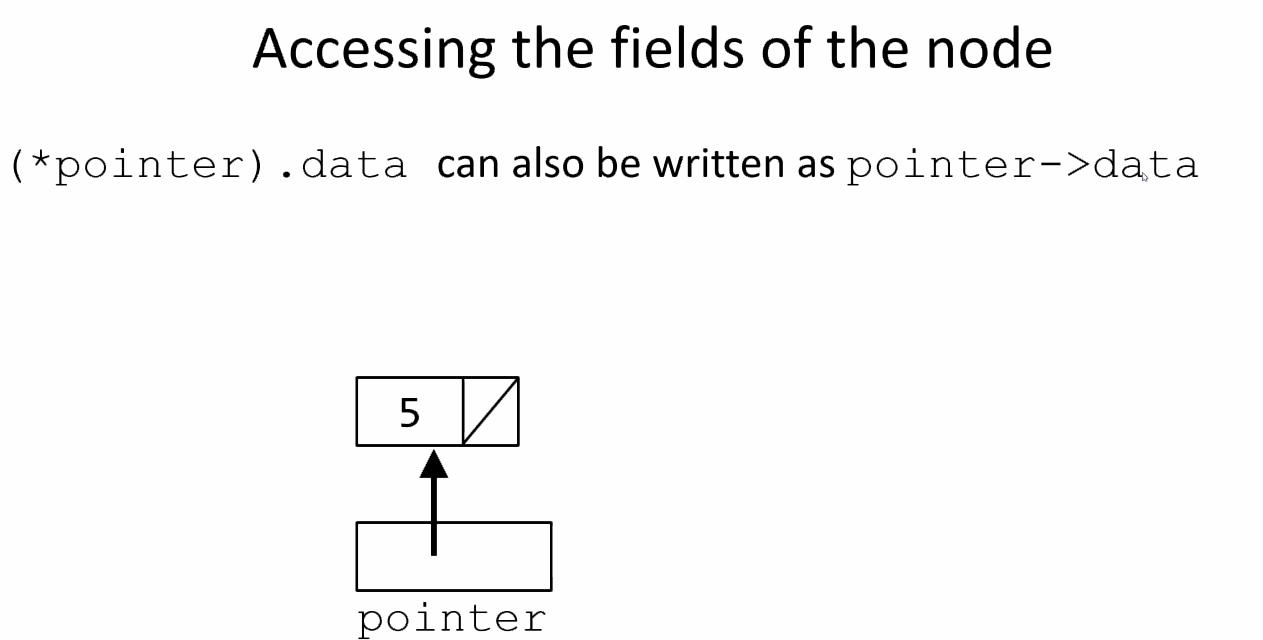
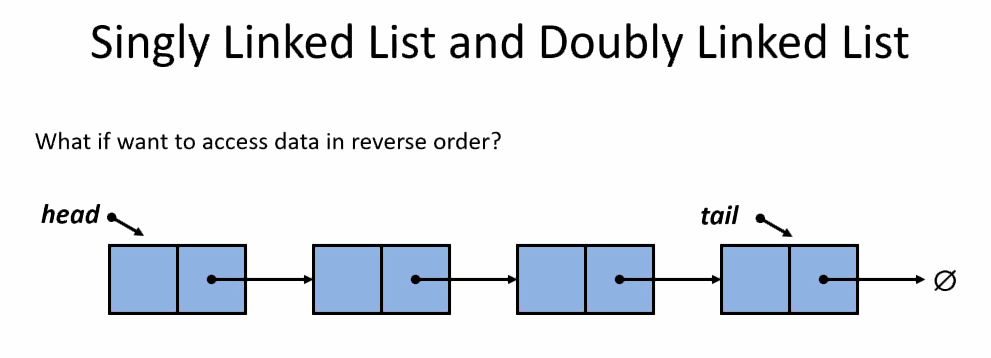
Lecture 7

9/16/2020

CPSC 131

1. Lecture
   1. Sequence Containers
      1. 
      2. Creating Matrix via vectors  
         
      3. This relates to amortized time in which the size of an array affects how much time it takes to process it
2. Singly Linked List
   1. 
      1. The difference between singly and doubly
   2. Implementations
      1. Singly Linked list implementation
      2. Doubly Linked List Implementation
         1. 
      3. Note, the implementation =/= the definition of them. You have to flesh them out via the header file
   3. Continuing on with Analysis of Vector Data type complexity
      1. There is an error on this slide. It has been changed from size () – same to size() – O(1)  
         
      2. 
      3. 
      4. 
      5. You will remember this by implementation in your code and flashcards
3. Singly Linked Lists
   1. 
      1. It is a node based data structure
      2. Each node consists of data and a next pointer
         1. Data = element
         2. Next = the pointer ->
      3. So like a class with two instance attributes
      4. The tail points to a NULL
      5. Size = number of elements
         1. Here, it is 4
      6. Sequence
         1. Head -> nodes -> Tail
   2. Inserting at the Head
      1. First, get a node!  
         
      2. Second, Make that new node point to the head   
         
      3. Third, update head to point to new node  
         
   3. Deleting at the Head
      1. Update head to point to the next node in the list  
         But, you have to assume that there is a temporary node that the pointer will point to  
         
      2. Delete the former first node  
         
   4. Insert at the Tail
      1. First, Always create a new node. Populate it  
         
      2. Second, have the new node point to whatever the tail node points to (namely nullptr)
      3. Relink the last pointer to point to the new node  
         
      4. Move the tail pointer  
         
      5. Note, allocate the new node and then set it to the slot in the list where it will fit in
   5. Draw data structure for this code  
      
      1. When it is created, it is empty
      2. Prepend (push\_front) = put in first element
      3. removeFront (pop\_front) = behead it
      4. ds.front() = show what is the first element in the list
   6. Draw data strcutrue for this code  
      
      1. Append (push\_back) = add a new element but do so at the tail end
      2. removeFront = behead it
      3. These are all part of sequence ctonainers. In other words, we tell the code where things are inserted/placed in a sequence.
   7. Nodes
      1. 
      2. The next pointer always points to a node
      3. Singly Linked list node  
         
         1. It is a class composed of Data\_t data, and Node\* next = nullptr
         2. Admittedly, we didn’t go over templates much in the previous lectures.
         3. The template allows you to genericize the structure. Hence the name template.
      4. Creating a node as a dynamic variable
         1. It is when a node is a point to other integers (???)
         2. 
         3. This returns a pointer to a freshly constructed object
      5. Accessing the fields of the node
         1. 
         2. IF you want to change the contents of the data
            1. Pointer-> = insert
            2. Change next
            3. Pointer -> = next
            4. Delete pointer
            5. Pointer -> delete

It deletes the node but leave behind a dangling pointer. SO you got have it point to null eventually.

* 1. Struct vs class
     1. No difference between the two.
     2. But the default access specifier is different between the two
     3. Struct default to public. Class defaults to private
  2. Creating a Linked List
     1. See slides for details
     2. Most importantly from Code to Picture
     3. 
        1. I don’t think he gave us the clear answer as he was testing us
  3. Singly Linked List Behaviors
     1. On cppreference.com , you can find a full list of behavrios that should be helpful for yoru projects.