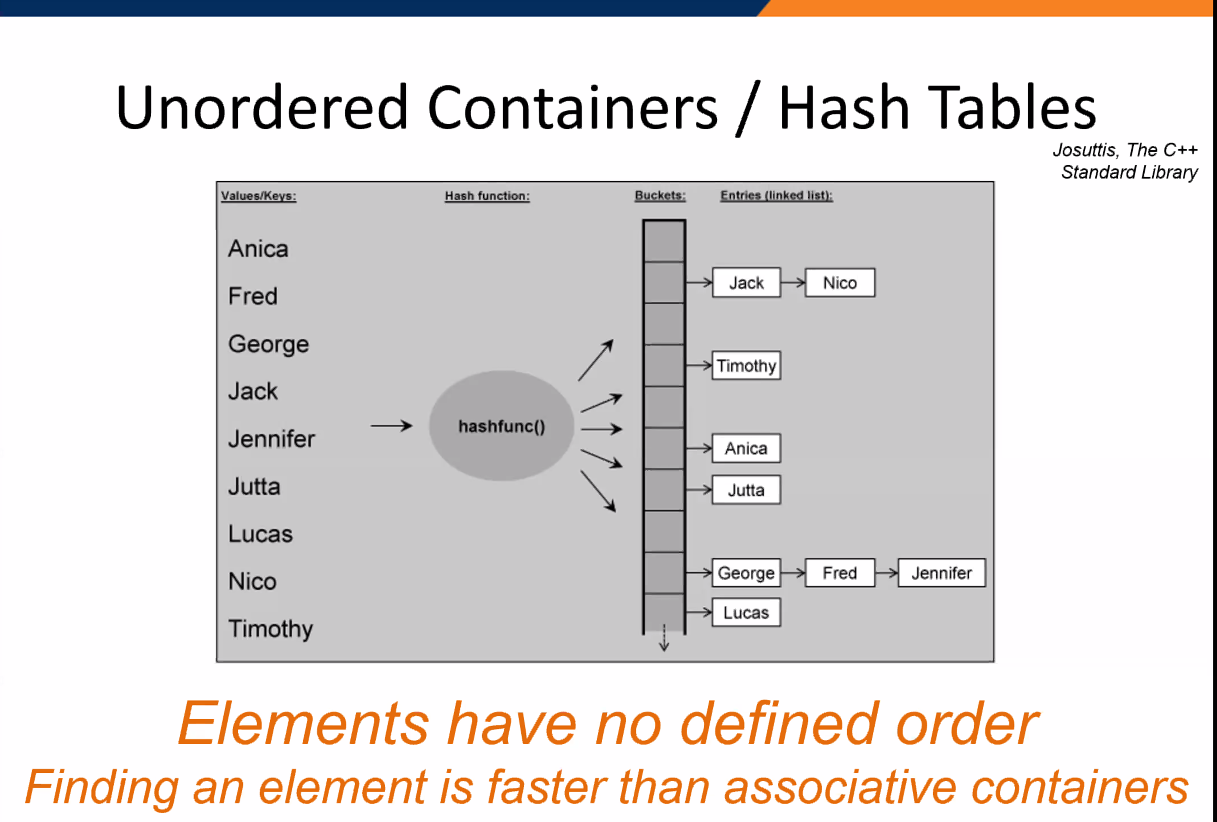
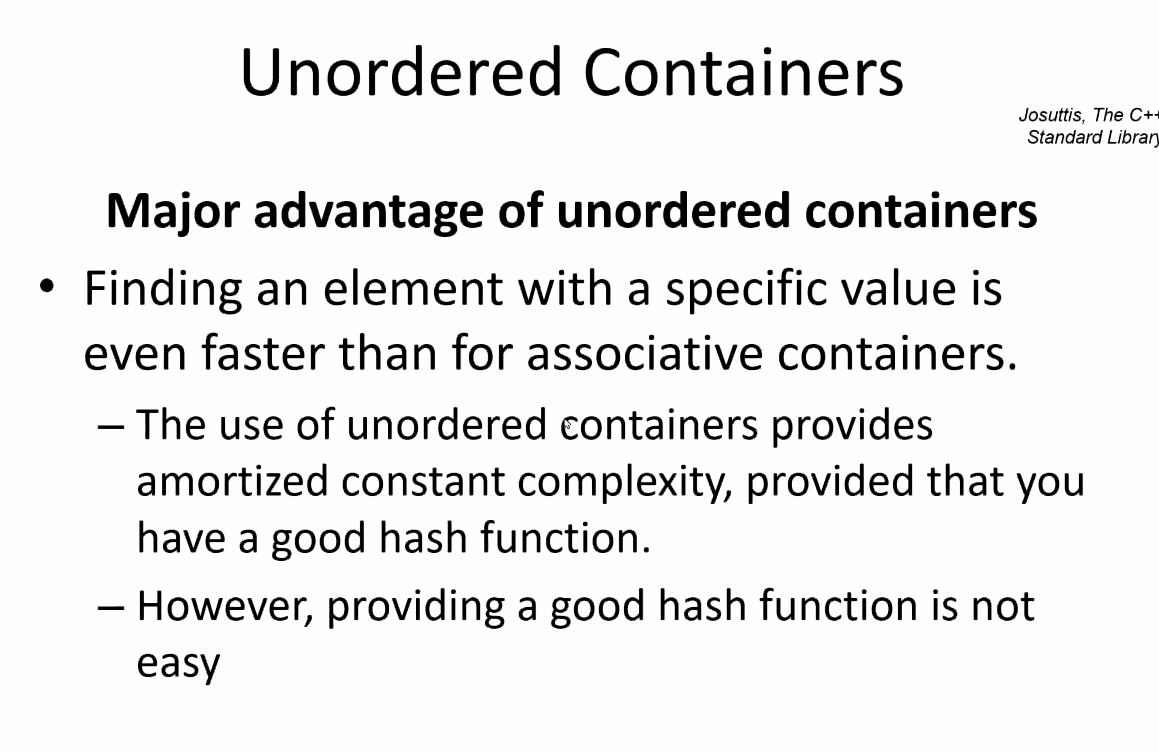
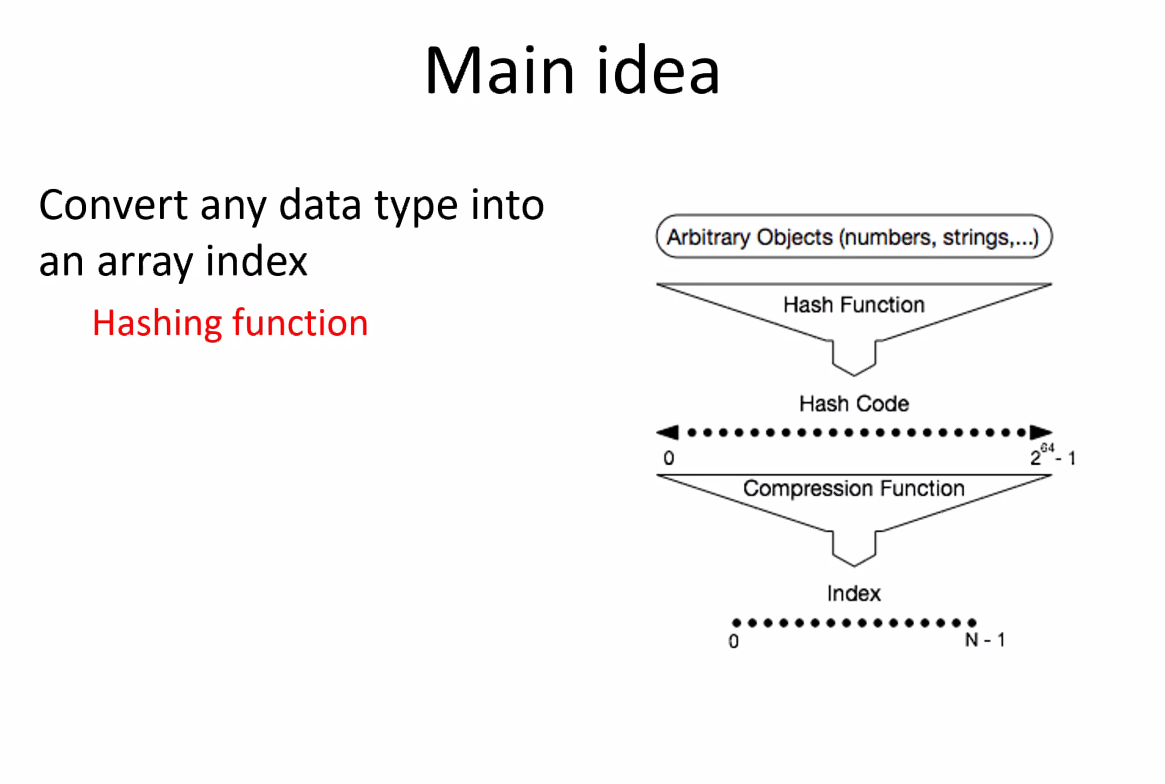
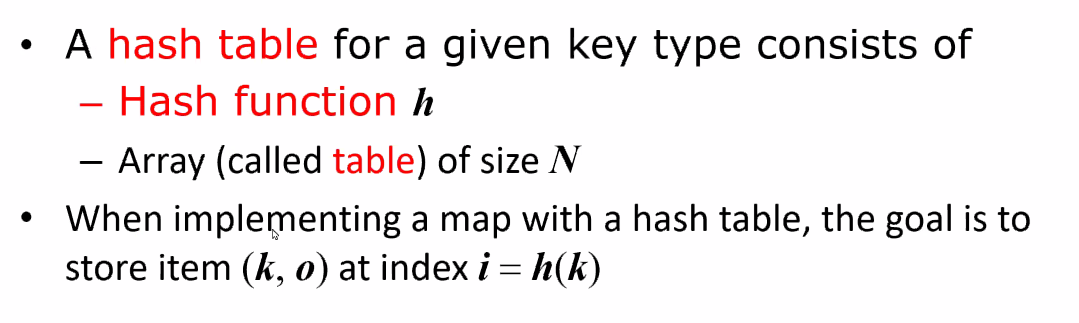
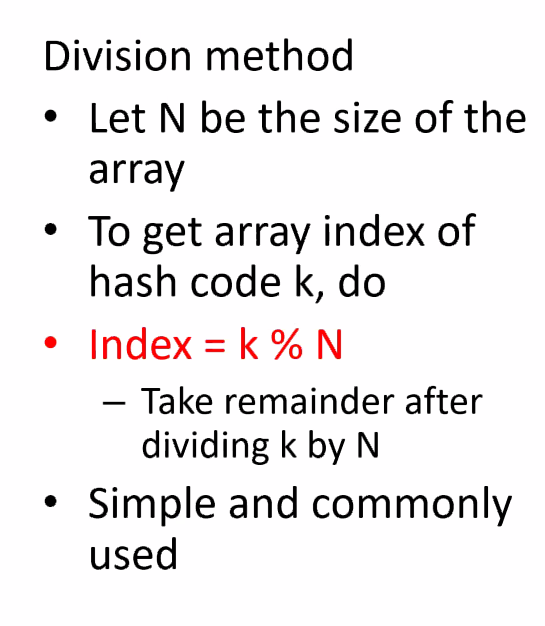
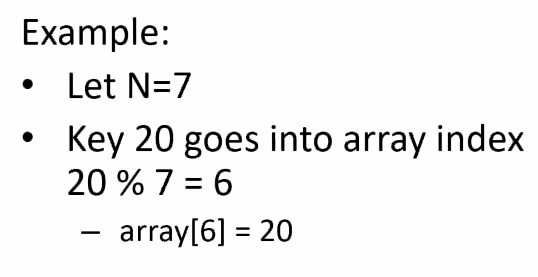
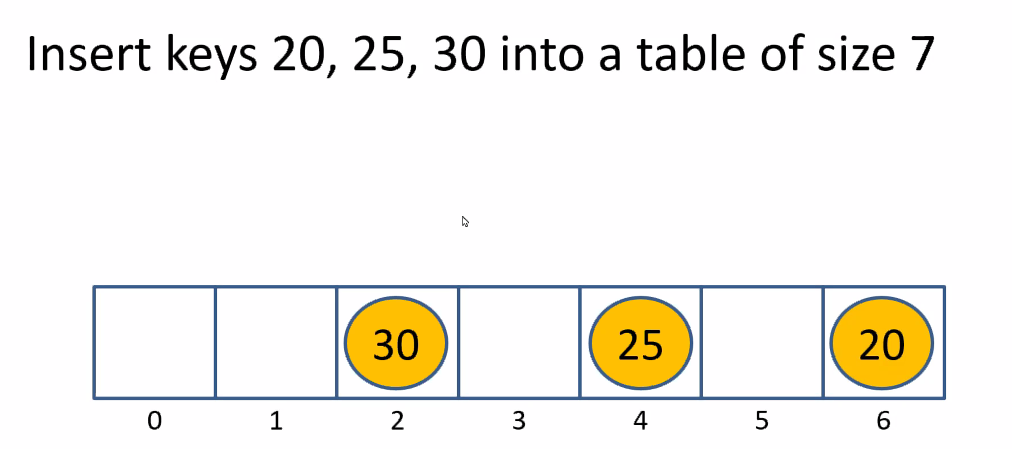
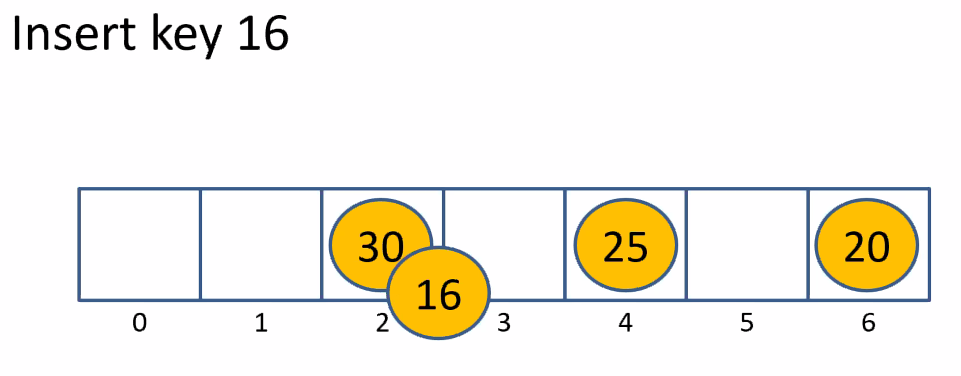
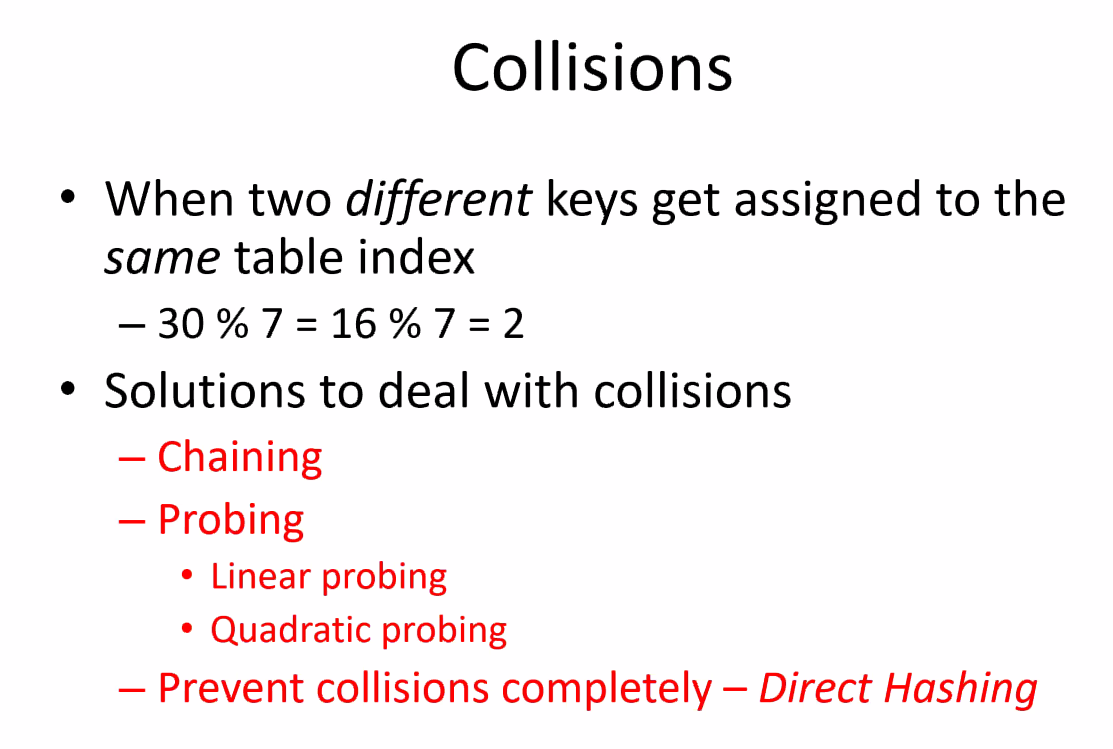
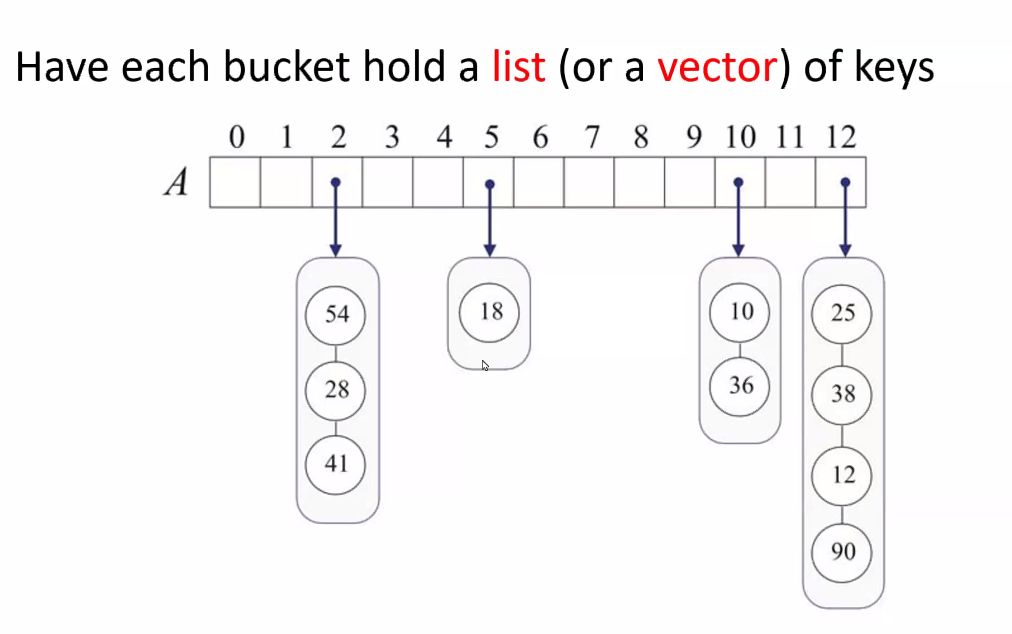
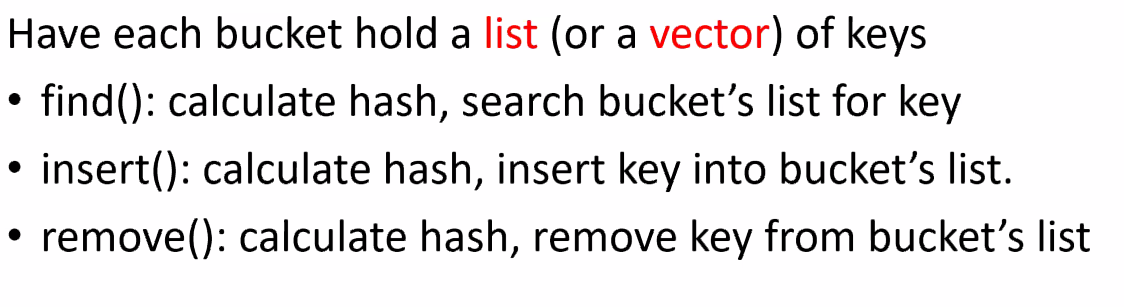
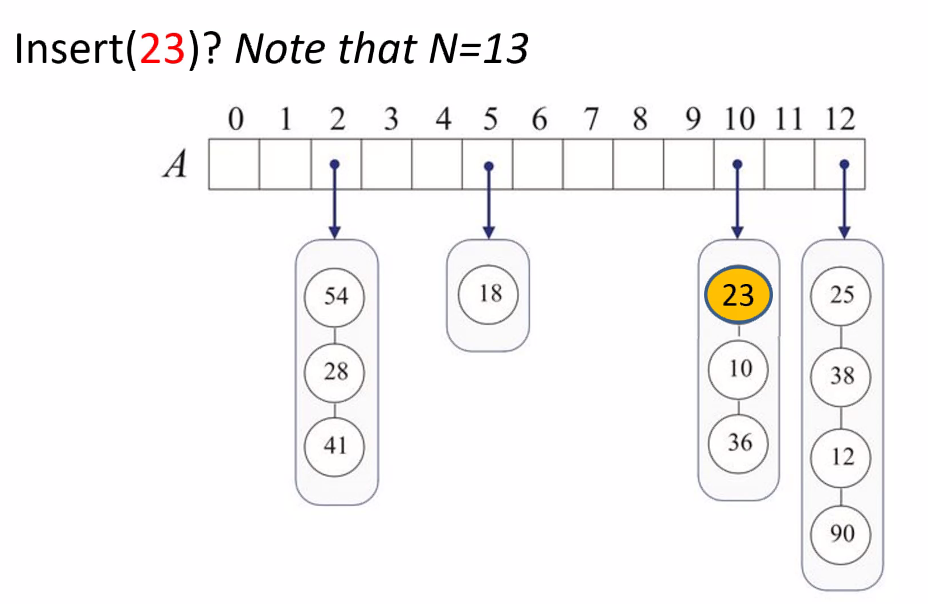
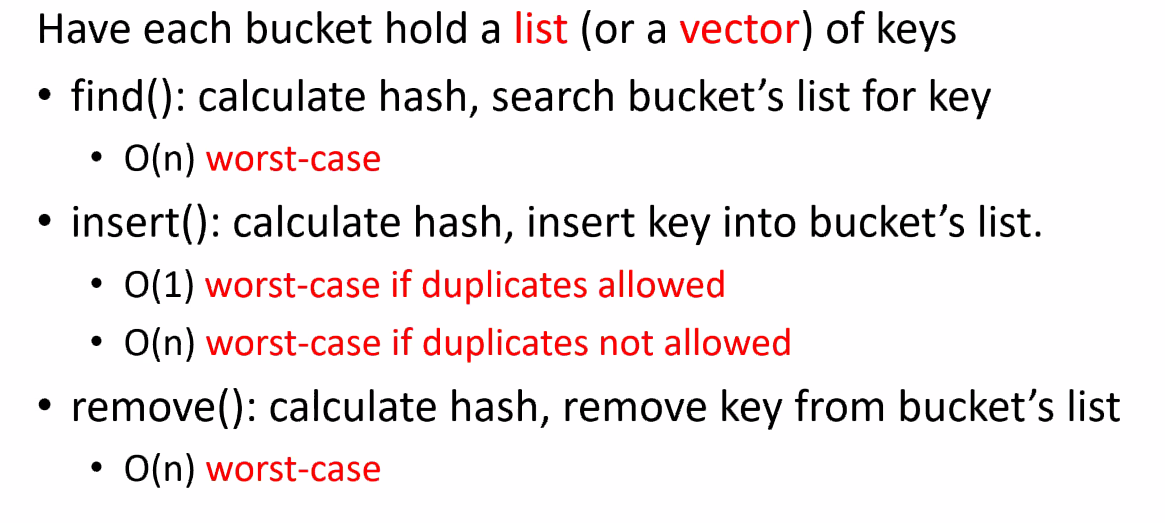
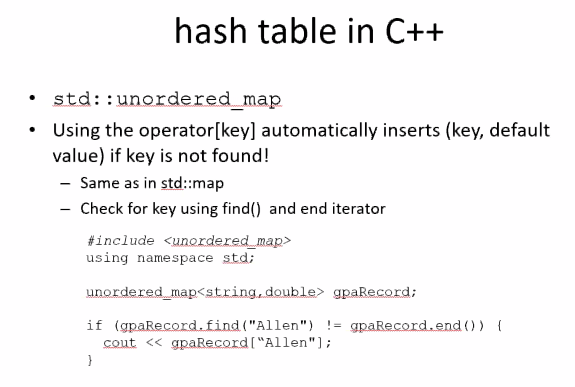
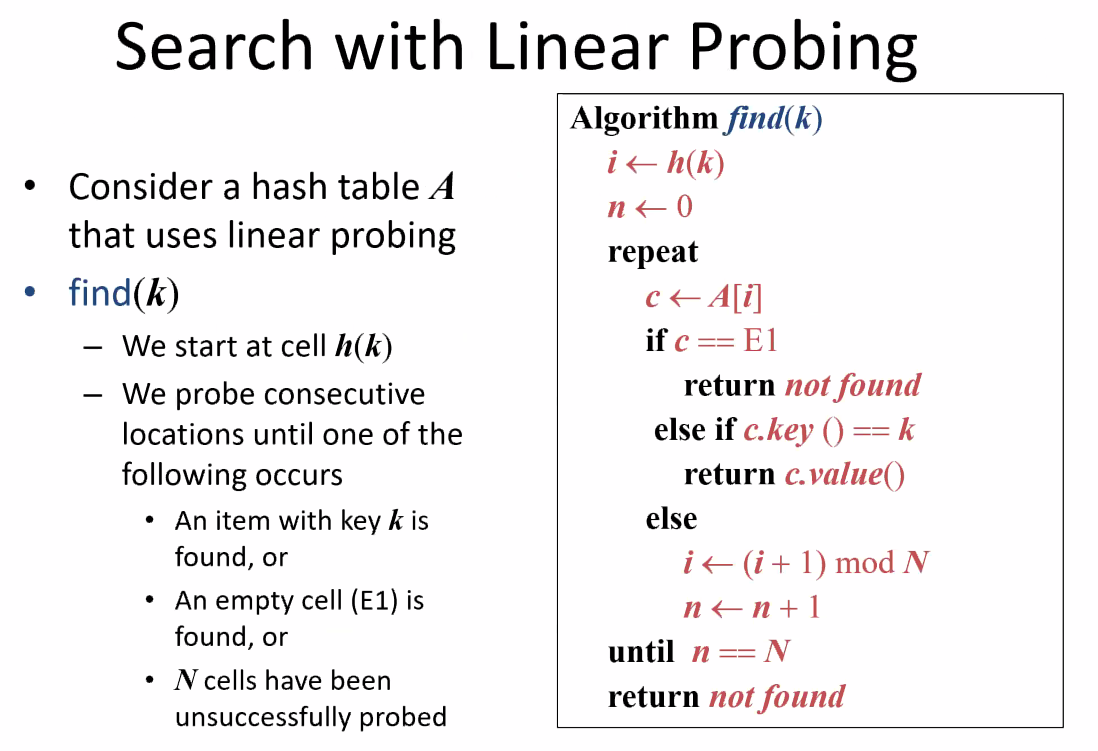
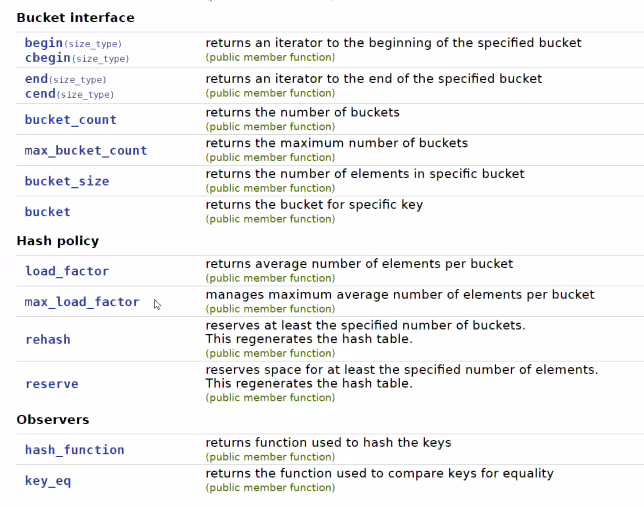
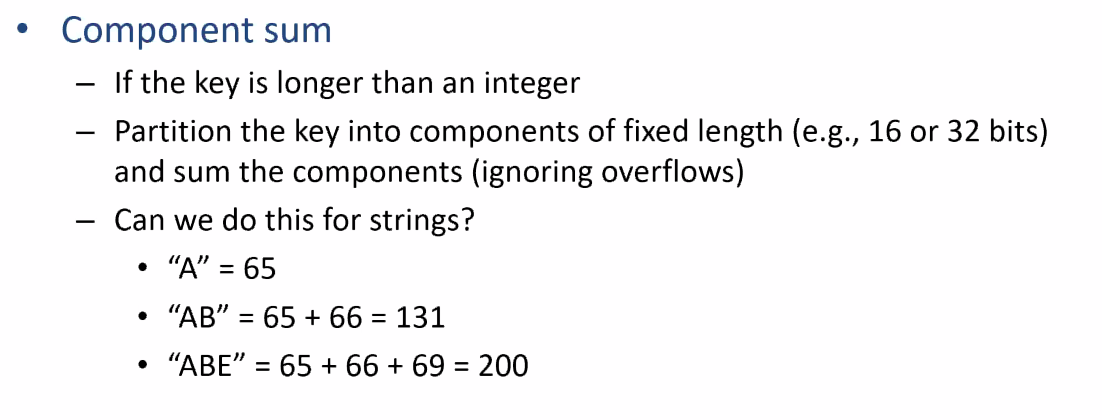
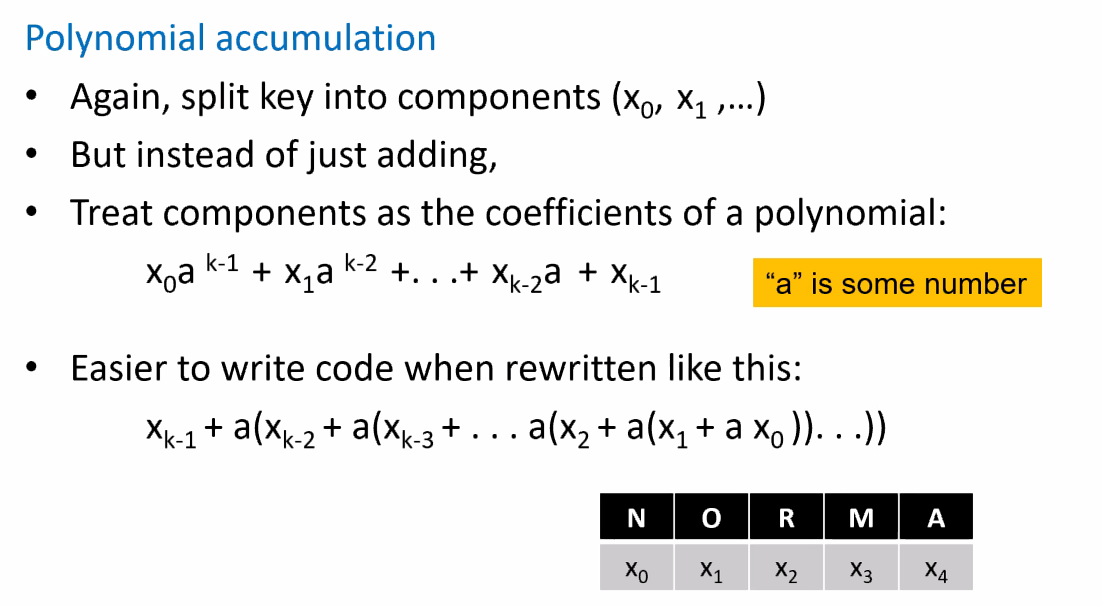
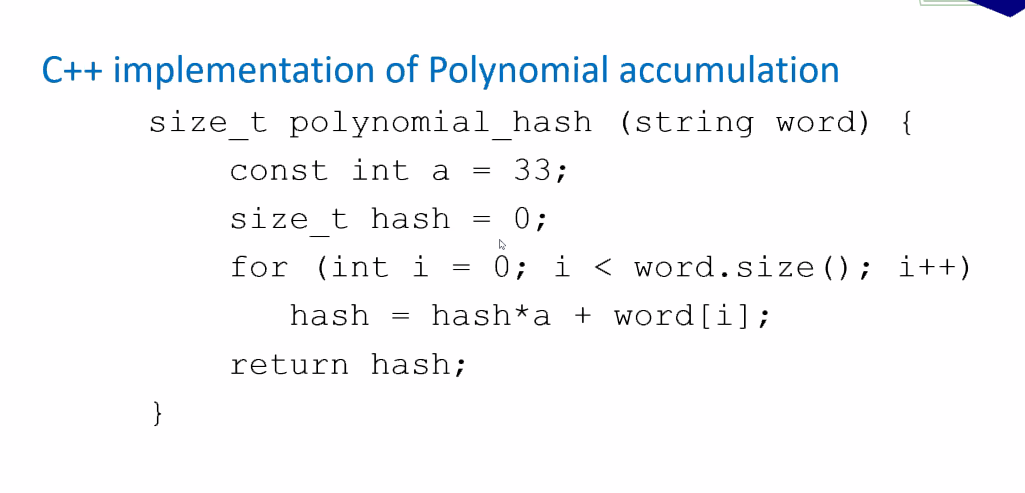
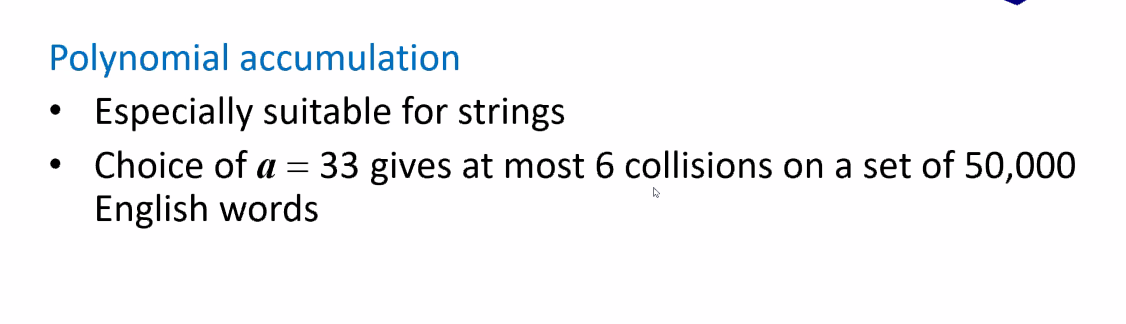
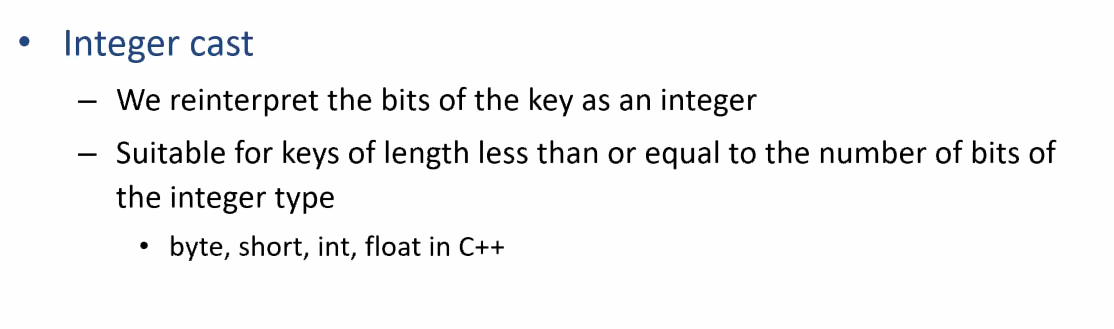
Lecture 24

CPSC 131

11/30/2020

1. Unordered Containers/Hash Tables
   1. 
   2. What’s in the bucket?
      1. A list of names
      2. Each element of the vector is a container/a list. So got to keep it small.
   3. Unordered Containers  
      
      1. Our hash table outperforms the other containers so far when it comes to keyless organization
      2. Every now and then when we insert, we have to rehash or resize the buckets
      3. Fastest performing thing for a majority of operations
      4. But in the case of constant insert/remove, use binary search tree instead
   4. Main idea  
      
      1. See below for what the compression function does/is in detail but, for now, it’s a way to control the size of the list using modulo
   5. Hash table
      1. 
   6. Compression Function
      1.  
      2. It’s modular math that can be used to create indices
      3. Example  
           
         n = 7  
         key 20 % 7 = 6  
           
         Here we have a **collision**
   7. Collisions
      1. 
      2. Chaining  
         
         1. We insert keys at the indices and form a list of keys
         2. No duplicates though
         3. 
         4. 
         5. 
      3. In C++  
         
      4. Linear Probing
      5. 
   8. References to study
      1. Functions  
         
      2. The OPP Book that came with the course
         1. It goes over the algorithms involved with hash tables
   9. Hash Codes
      1. 
      2. 
      3. 
      4. 
      5. 
2. Homework
   1. Look for a unique hash code