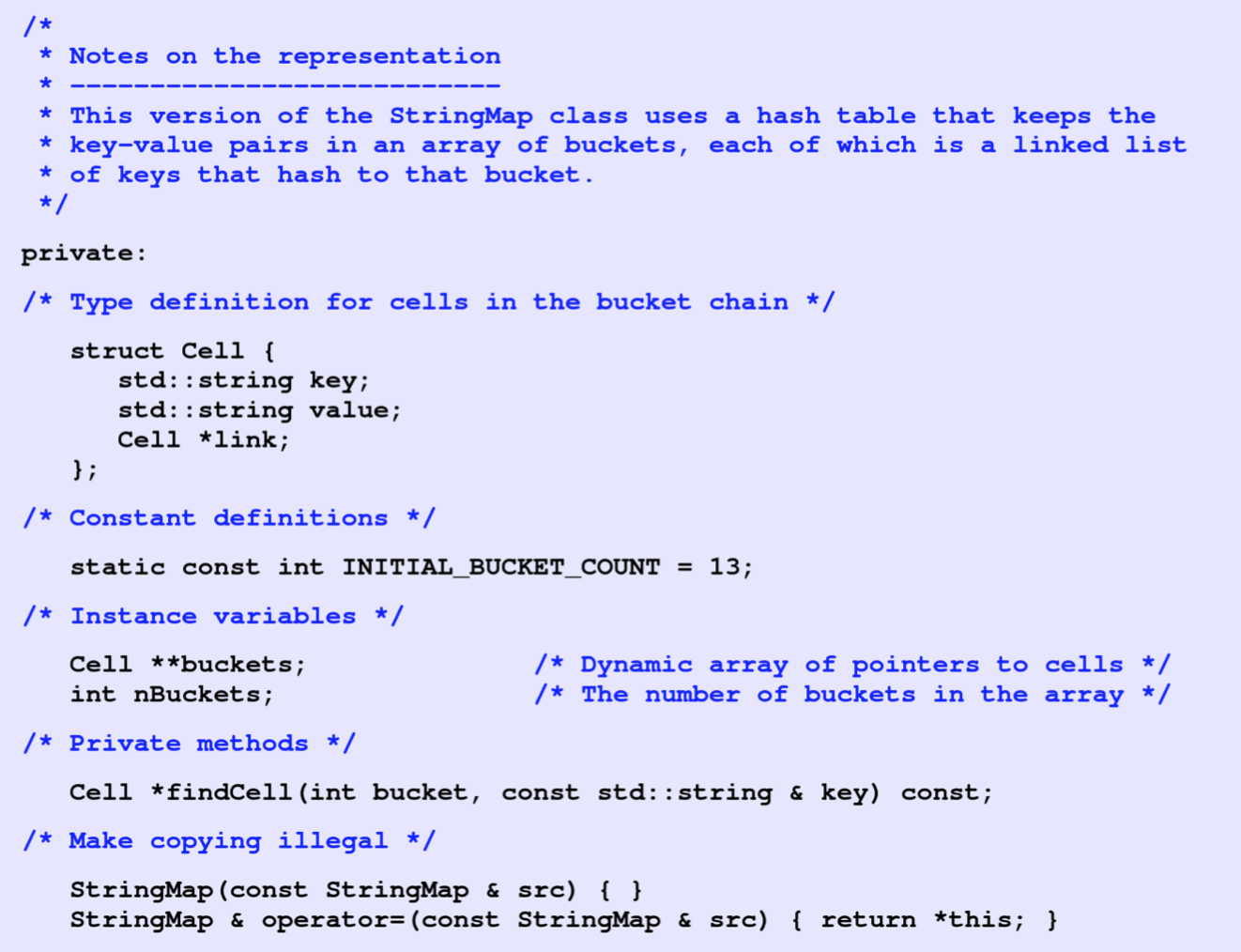
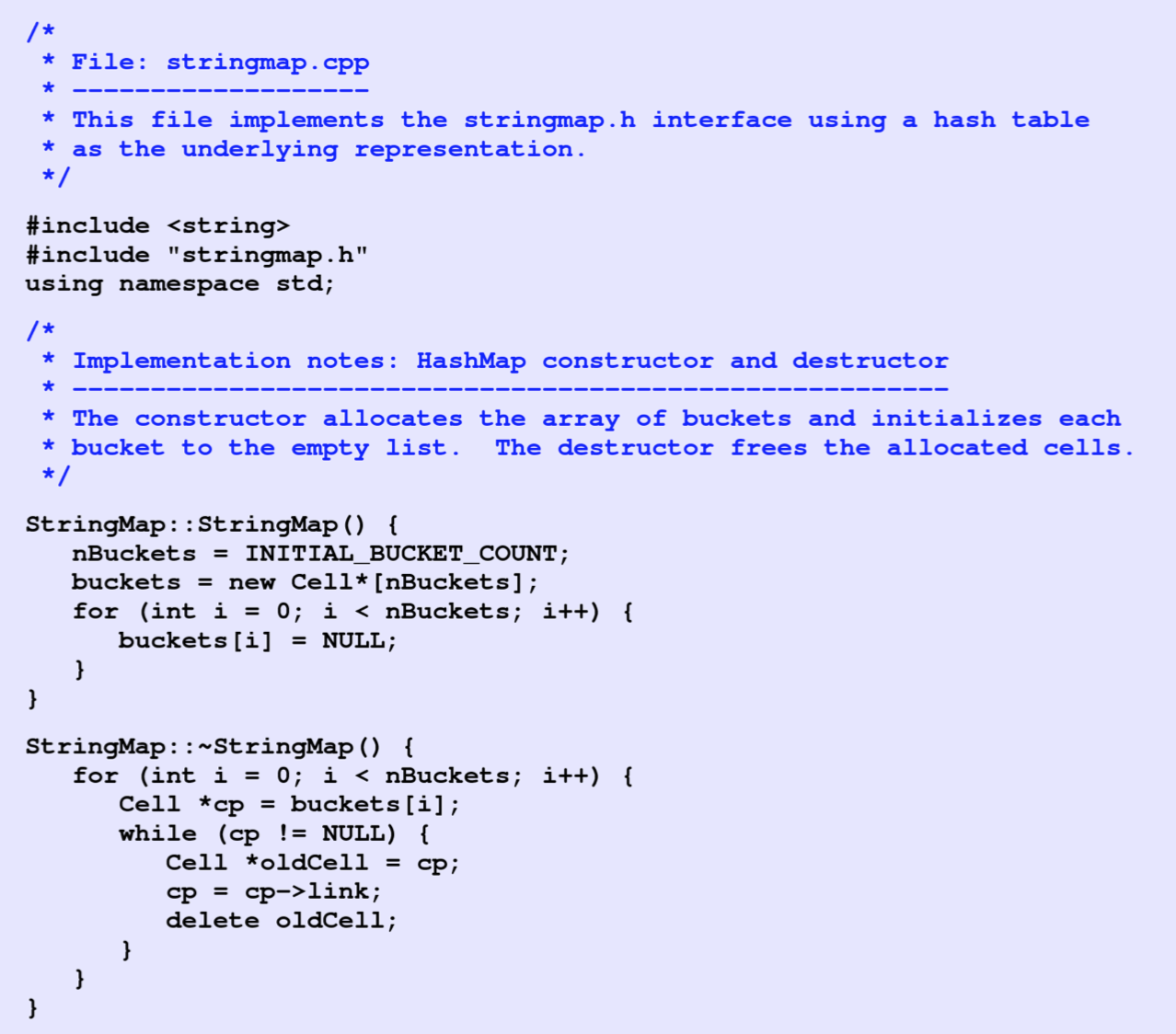
**HashMap**

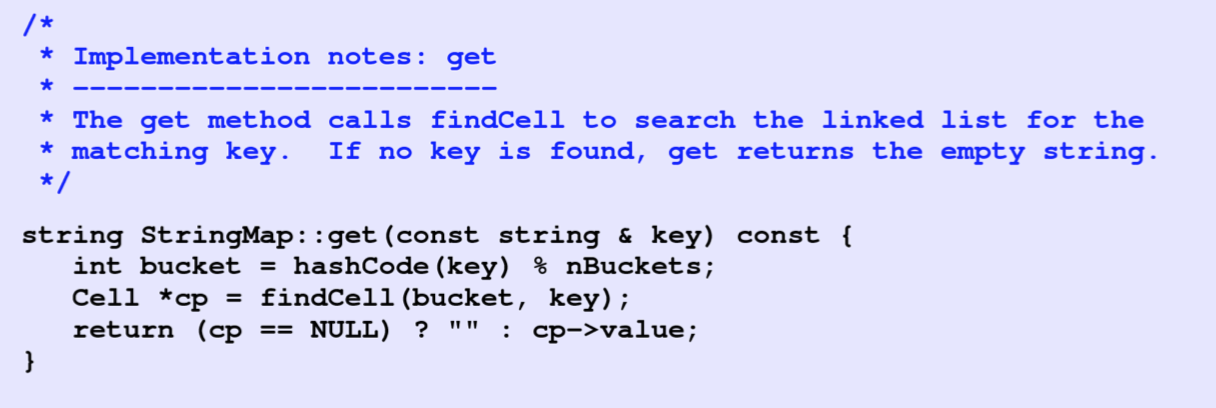
**CS 300/CS 500: Advanced OO Programming in C++**

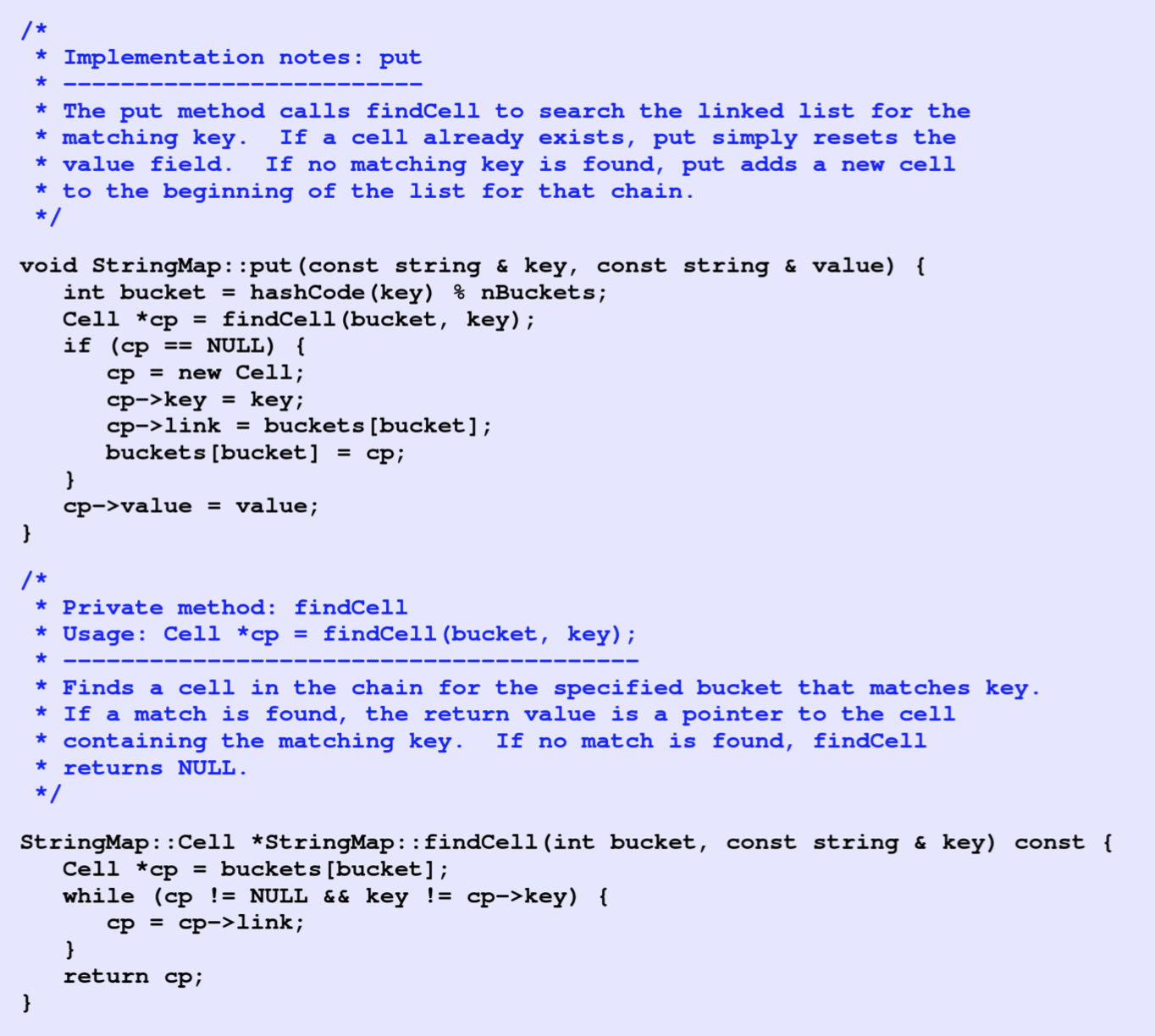
In this activity, you will review a hash table implementation of StringMap and then transform it into a template version using const correctness.

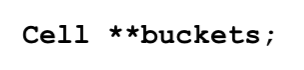
Please work in groups of 2-3. Start by reviewing the hashtable implementation of StringMap below. Please submit the word document and any .h/.cpp files you write at the end of class (one per group). Whatever you have completed at the end of class is sufficient.









1. What is **nBuckets** and why is it necessary?  
     
   **That is the numbers of hashmap buckets in use. It is necessary to keep track of the number of buckets to evaluate potential for collisions, and also to use when calculating the hash.**
2. Notice the declaration of **buckets  
   **In previous problems, we have used dynamic arrays where we have used a reference to hold a pointer to the base type. Explain why this declaration requires two asterisks.

**buckets is a list of Cell pointers, or a pointer (to the list) to a pointer of Cells.**

1. Convert the StringMap code above to a template HashMap (KeyType, ValueType) class using const correctness. We are missing a few methods. Add the methods size, isEmpty, constainsKey, remove, and clear.

1. Ensure the HashMap supports deep copying.
2. Only do this problem if you have time. There is some subtlety required to implement the hashCode method for different datatypes. If the datatype supports a toString method, you could convert the object to a string and pass it to the hashCode method in the slides. For integers, you could return the key and the bitwise & with the HASH\_MASK. Writing good hash functions is complex.