* Final
  + 4-5 questions on the final. Probably five. Very similar to the midterm
  + Wed 12-150pm
* Concordance
  + Agreement
  + List of important words (index). Back of the book
* Multimaps
  + This is one of the study guide problems that he assigned
  + Create a program that reads in an file and creates a multimap that is able to read in each word and indexes which line the word is on
  + Input
    - "A list of words - 1
    - in tihs file of - 2
    - words to test - 3
    - Concordance. - 4
    - More words. " - 5
    - A: 1, …. words: 1 3 5
  + Class Concordance
    - Typdef multimap<string, int> wordDictType;
    - Public:
      * Void addWords(string, int)
      * Void readText(istream &)
      * Void printConcordance(ostream &)
    - Private:
      * WordDictType wordMap;
  + ReadTest
    - Void concordance:: readText(istream & in)
      * String line;
      * For (int I = 0; getline(in, line); I++)
        + AllLower(line)
        + List<string> words;
        + Split(line, " ,.;:", words)
        + List<string>::iterator wptr;
        + For(wptr = words.begin();wptr != words.end(); wptr++)

AddWord(\*wptr, I);

* + Add Word
    - Void concordance :: addWord(string word, int line)
      * WordDictType::iterator low = wordMap.lower\_bound(word);
      * WordDictType::iterator high = wordMap.upper\_bound(word);
      * For(; low != high; ++low)
        + If((\*low).second == line)

Return; //word already appeared on this line

* + - * + //didnt occur so add now
      * WordMap.insert(wordDictType::value\_type(word, line));
      * // or wordMap.insert(Pair(word,line));
  + Print
    - Void concordance::printConcordance(ostream & out)
      * String lastWord = "";
      * WordDictType::iterator pairPtr;
      * WordDictType::iterator stop = wordMap.end();
      * For(pairPtr = wordMap.begin(); pairPrt != stop; ++pairPtr)
        + If(lastword == (\*pairPtr).first;

Out << " " << (\*pairPtr).second;

* + - * + Else

Lastword = (\*pairPtr).first;

Cout << endl << lastword << ": " << (\*pairPtr).second;

* + - * Cout << endl;
  + Split – possibly on the final
    - Void split(const string & str, const string & delim, list<string> & words)
      * String temp = "";
      * For(string::const\_iterator I = str.begin(); I !=str.end(); ++I)
        + If(delim.find(\*I) == -1)

Temp += \*I;

* + - * + Else

If Temp != ""

Words.push\_back(temp);

Tmp = "";

* + - * If (temp != "")
        + Words.push\_back(temp);
  + All lower
    - Void alllower(string & str)
      * For int I = 0; I < str.length(); I++
        + If str[I] >= 'A' and str[I] <= 'Z'

Str[I] = str[I] -'A' + 'a';

* + Map<string, list<int>>
* Number 2 Page 405 – Use composition instead of heritance.
  + Template <class T>
  + Class Map //remove the inherited public set class
  + Public:
    - .
    - .
  + Private:
    - Set<T> parent; // use set as a private variable
* Another question (didn’t catch the number)
* Make a tree a binear search tree by making the leftChild point to the eldest child and the rightChild point to the siblings
* Class Nde
  + Node \* eldest\_child;
  + Node \* next\_sibling;
  + T value;
  + Bool parent. (false if there is a sibling and true if its pointing to the parent)
* To find all children. Go to the eldest child. Then loop till the sibling node is null and print each sibling.
* To find the parent. Change the youngest siblings null pointer to the parent