CS 251
Spring 2009
Voicu Popescu
Assignment 4
Due Friday February 13, 11:59PM

Vectors, Lists, and Sequences

1. Implement a C++ class that models a collection of words with an array of lists. A list stores words. There is one list of words for each array element. Each array element corresponds to a lower case letter and the list of the array element stores all words in the collection that begin with that letter. Words are spelled exclusively with lower case letters. The word collection can store duplicates. The data structure should not make any assumption about the maximum number of words. The maximum length of a word is 20 characters (i.e. letters). The word characters are stored in a null terminated character string (i.e. there is a 0 after the last character of the word; a 0 and not a ‘0’). Your data structure should provide:
    a. A **constructor** that makes an empty word collection (array elements have empty lists).
    b. An **insert** method that adds a given word to the word collection in constant time.
    c. An **isPresent** method that output “Yes” to standard output if and only if a given word is already part of the word collection, “No” otherwise; running time $O(n_l)$ where n_l is the length of the longest list.
    d. A **remove** method that removes one occurrence of a given word, if any; running time $O(n_l)$, as before.
    e. A **removeAll** method that removes all occurrences of a given word, if any; running time $O(n_l)$, as before.
    f. A **removeDuplicates** method that eliminates all but one occurrences for each word in a collection.
    g. A **print** method that takes a character input, and output all words starting with that character.
    h. A **printAll** method that output all lists, alphabetically (First the list for character a, then b, then c and so on. The list for each single character may be not alphabetically sorted in itself).

2. You notice that many words begin with the letter *‘p’*, which makes the list for *p* long and the **isPresent** method inefficient. Describe a modification to your data structure that alleviates this problem, and give a pseudocode description of the **insert** and **isPresent** methods for the modified data structure.

3. Extra-credit (3%): Problem C-5.6, page 247 of text book.

4. Turn in instructions
    a. Question 1: Name your program file as “words.cpp”. “words.cpp” should implement the operations given in “words.h”. You may add some code to “words.h”, in fact you should. However, do not change the names and signatures of the given operations. A test file “wordsTest.cpp” is also provided to test the data structure. Examine and run that program to test your operations. You will not turnin the test file.

    b. Question 2: Turn in a PDF document named as “a4.pdf”. This file should include your description of the modification and pseudocode descriptions of the **insert** and **isPresent** methods for the modified data structure. **Read General turnin instructions parts E, F and G very carefully, on the website.**

c. Question 3 (Extra): Turn in a PDF document named as “a4extra.pdf”. This file should include any explanations and the proof. **Read General turnin instructions parts E, F and G very carefully, on the website.**

d. Your codes MUST compile on lore.cs.purdue.edu for it to be graded.

e. All of your documents (words.h, words.cpp, a4.pdf, a4extra.pdf (OPTIONAL)) MUST be in the directory A4.

   In the directory above A4, run the command on lore.cs.purdue.edu

   ```
   turnin -c cs251 -p A4 A4
   ```

   optionally run the following command to verify what files were submitted

   ```
   turnin -c cs251 -p A4 –v
   ```

   Warning: turnin will be automatically disabled at 02/13/2009 at 11:59 PM EST and you will be unable to turnin your assignment after this time.

   Warning: turning in multiple times is ok but only the last one will be graded as each turnin permanently overwrites the previous one.

   Warning: failure to turnin exactly according to the instructions given above will result in your A4 receiving a grade of 0.