1. Introduction

The purpose of this laboratory is to develop a program that reports if two words are anagrams, and displays the permutations of the characters of both words being examined as anagrams. Use of pointers is emphasized in this laboratory project. Use of objects from user-defined classes are also encouraged.

2. Task 1

Create a program that fulfills the following requirements.

3. Requirements

3.1 Functional Requirements

1. The program shall display the letters of two words and whether they are anagrams of each other.
   
   1.1. The two words (lowercase letters only) to be evaluated as anagrams shall be provided to the program as arguments on the command line. *(Using a command line is optional.)*

   1.2. The display format shall consist of a statement whether the two input words are anagrams, and display of each word with characters arranged in ascending alphabetic order.

   Format (example for two words *(silent and listen)* that are anagrams):

   \[
   \begin{align*}
   \text{silent} & \quad \text{and} \quad \text{listen} \quad \text{are anagrams} \\
   \text{------} & \quad \text{------} \\
   \text{eilnst} & \quad \text{eilnst}
   \end{align*}
   \]

   Format (example for two words *(apples and oranges)* that are NOT anagrams):

   \[
   \begin{align*}
   \text{apples} & \quad \text{and} \quad \text{oranges} \quad \text{are not anagrams} \\
   \text{------} & \quad \text{------} \\
   \text{aelpss} & \quad \text{aegnors}
   \end{align*}
   \]
3.2 Performance Requirements

None

3.3 Implementation Requirements

1. A class called Anagram shall be used to define two objects, one for each input word. The class shall provide functions to get a word from the command line, display the word, determine if the word is an anagram of each other word on the command line, and display the program output. (Using a command line is optional.)

2. The data structures needed to compare and permute the words shall be character arrays.

3. Dr. Carter’s suggested approach: All access (reading and writing) to the character arrays shall be in pointer format, not using '[]'.

4. Dr. Carter’s suggested approach: All function parameters and calling arguments shall use pointer format where addresses need to be passed to a function. One exception to this is when the entire array is to be passed to a function in which case just the array name is sufficient.

5. Dr. Carter’s suggested approach: My thought is to use the pointer notation for accessing the array. For example, let word 1 be represented by array1 and word 2 by array2. Each is an array of 26 integers. Each element corresponds to a letter of the alphabet (lower case). Thus, (*array1) + c - 'a' is the 3rd element of abc which corresponds to 'c'. So when a letter ch is obtained from word 1, just increment ((*array1) + ch - 'a')++.

6. Dr. Carter’s suggested approach: Then compare the count between the same locations from beginning to the end of both array1 and array2. If they match for all elements the words are anagrams.

7. If you wish, you can use an approach other than the suggested by Dr. Carter.

8. The course style guide shall be followed. In particular, satisfy the style guide for classes, objects, pointers, and arrays.

4. Submission

If you are submitting your lab results in hardcopy, submit all code files (at least six files - .cpp/h for the main program and for each of the classes) for this task that implement a complete and correct program. Also submit an image of the display window.

If instead your section uses an electronic archive to submit programs for grading, please submit your source files to the archive where they will be compiled and executed several times with different test input text files.