Nathan Reichert

Saint Cloud State University

CSCI 301

Project 1

Anagram Checker

Due: 2 September, 2015

Nathan Reichert

Project 1-Anagram Checker

Due 2 September, 2015

Design Document

To begin, this program was created to check character arrays created from input data for anagrams. Anagrams are strings that contain the same characters as each other, but in any order. This particular program obtains two lines of input from the user, counts the letters once for the first line of input, and once again for the second line of input, and then compares the counts for each instance to determine whether or not the two lines of input are, in fact, anagrams. In this case, only letters are of importance to the anagram checker, and will ignore special characters, numbers, and white spaces. The maximum number of characters per input line is 60.

This program uses character arrays and integer arrays. The character arrays are used to hold the characters from the input lines. The integer arrays are used to hold the number of occurrences of each letter of the alphabet, counted from each of the character arrays.

The most prominent function used in this program is from the standard library and gets the entire line of input from the user, and is able to assign it to a character array. This is the cin.getline() function. There are other defined functions written into this program that count the occurrence of each letter of the alphabet contained in each of the character arrays, called countLetters(), as well as a function to compare the letters counted as integer arrays, called isAnagram().

The main program prompts the user for a first line of input, then assigns that to a character array, and the letter counting function is called. Then, the user is prompted for a second line of input, which is assigned to a second character array, and a second instance of the letter counting function is called. The letter counts are then compared to see if they are indeed anagrams, and the corresponding result is displayed for the user.

Nathan Reichert

Project 1-Anagram Checker

Due 2 September, 2015

User Document

Anagrams are instances of characters that have the same number of occurrences, but the order is not important. The program, named ‘anagram.cpp’, detects whether or not two lines of input are anagrams. The program gets two lines of input from the user, one line at a time, which must be 60 characters at maximum. After this, the program counts the letters in each line of input, compares the letter frequencies, and outputs whether or not the two lines are anagrams.

To compile the program in the CSCI2 unix terminal, enter: g++ anagram.cpp

To run the program, after compiling, in the unix terminal, enter: a.out

The program will then prompt the user to enter a line of input (60 characters max).

For example: Angered paternal orchestra

The program will then prompt the user for a second line of input (also 60 characters max).

For example: Parental carthorse enraged

After the second line of input is entered, the program will run, and return whether or not the two lines entered are actually anagrams.

For example: These are definitely anagrams.

Please note that the program is only concerned with the letters in strings, meaning that any special characters, numbers, and white spaces will be completely ignored in terms of anagram status.

Nathan Reichert

Project 1-Anagram Checker

Due 2 September, 2015

In summation, this program utilized a standard library function to assign input lines to character arrays, integer arrays to store letter counts, and written functions to count letters and compare arrays. Also, the basic forms of input and output were vital for program success and user interface.

One specific piece of knowledge gained from the creation of this program corresponds to character arrays and their initialization. For example *char charray[size];* will initialize a character array, but it was examined that *char charray[size]={};* will initialize a character array, and ensure that each index is specifically an empty string.

It would be suggested that the Big-O of this particular program is linear. If the number of characters in the two input lines is n, the Big-O would be O(n). This is because each of the characters are only read once, counted, and then compared, with no type of sorting necessary.

It is logical to state that there are many ways to accomplish the goal of anagram checking. Another potential method for completing such a goal, aside from the one presented in this anagram.cpp program, would be to get the lower case version of each letter character into a character array (ignoring numbers, special characters, and white spaces), then to sort each of the character arrays, and then to directly compare them to see if they are the same. The sorting algorithms, depending on how they are implemented, would then affect the Big-O of the program.