Name Beteab Gebru;

Date: 01/16/2017

Project one: **DETECTING PALINDROMES**

**Introduction**

*Palindromes are phrases, sentences or any sequence of characters that are equivalent when read right to left and or left to right. In c++ we can detect such structures by parsing through the array of characters in the string and doing appropriate comparison.*

**Data Structures**

The program uses a **string** called **phrase** to store the user input and an **int** called **phrase\_length** to store the size of the array holding the string characters. In addition to these I use various loop control variables that with limited scope. The user input function is such that string will only store non-empty alphanumeric characters. The program is built to prompt the user for input until appropriate input string is keyed in.

**Functions**

The program calls various predefined functions from the library as well as functions defined by me the programmer.

::the following are user defined functions;

-void user\_input(string&, int&);//to retrieve user input

-bool is\_palindrome(string&, int&);//check the input is a palindrome and returns bool

-bool myassert(string& , int& );//to assert user input is workable and returns bool

::function calls from **c++** library are as follows;

-getline(std::cin, string);//to capture whole line of input

-string.size();//to retrieve array size for the string(array of characters)

-putchar(tolower(string));//to convert chracters into lower-case letters

-isalnum(string[x])//to check if a character is alpha-numeric (depends on locale)

::the following c libraries were called

-#include <cstring>

-#include <iostream>

-#include <locale>

-using namespace std;

**The Main Program**

Phrase and phrase\_length are declared and initialized in the main function and then have reference to them sent over to *user\_input()* function. In the *user\_input()* function user is prompted for input value and stored; then *myassert()* function is passed reference to the string to assert the user input validity(invalid if string is NULL string or entirely nonalpha-numeric). If *myassert()* returns false user is prompted again for input until valid input is entered. Once we assert user input validity the characters are all converted into lower case for effective comparison. The user input is sent to the *is\_palindrome()* function to see if palindrome and appropriate message is displayed.

**Conclusion:**

Alternative method: We could copy the string elements that are alphanumeric from given input string and do *is\_palindrome()?* Test but that would waste memory space. This program is able to pass by reference and therefore is able to work with the original address and just read the right charcters to compare. Ex; the loop skips through invalid chars found in strings to find the next valid char to compare to. Saves time and space by using this method. It only compares ½ n times(n=character array size) since the loop variables stop at midpoint of the given array of characters.