Name: Beteab Gebru

Due Date: Feb 09 2017

CSCI 301: Section 02

CSCI ID: cs301154

**Project 3: Program Documentation**

**Introduction**

This 3-part project programs that use recursion to solve problems are as follows.

* **Finding GCD**

Finding a GCD for two numbers is a recursive process that takes remainders of n1%n2 until the remainder is zero upon which we can say we have found the GCD of n1 & n2.

Ex. {GCD(n1,n2)🡪GCD(n1,n1%n2)🡪GCD(n1,(n1%(n1%n2)))………. And so on.

**🡪 Prime factorization**

The program takes a number and initially checks is it if a prime number. If it is not prime number, it goes on to find every factor starting at 2 and ending at number-1. It achieves this by constantly calling itself with number and a new divisor to check if it is a factor. Once divisor is found to be a factor it is printed after checking its primality.

.

* **Recursive Palindrome checker**

. The palindrome program uses the same techniques for capturing the input, however the bool **is\_palindrome()** function continuously compares the two array locations at time. Until sentence [a] != sentence [b] occurs the function calls itself with a++ and b-- parameters until a=>b

**Data structures**

Finding GCD:

🡪the program uses int variables to store the input integers operates on the numbers until gcd is found(when remainder is 0).

Prime Factorization:

🡪this program uses integer variable to store user input and goes on to find the prime factors of it and stores the as int variables.

Recursive palindrome checker:

String,int,

**Functions**

**Finding GCD**

-simple program with only one additional function to the main().

int gcd(int n1, int n2), takes in two integers and returns their greatest common factor.

**Prime Factorization:**

Program has two user defined functions apart from the main() function.

-bool Is Prime(int number)//returns true if user input is prime number

- void Prime Factors(int number, int divisor)//takes in user input and finds the factors after which the factor is checked for primality.

**Palindrome detection**

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 programs::**

The programs are supposed to be more elegant versions but not necessarily efficient in the case of the palindrome detection.

**Conclusion**

The recursive implementation of these three programs does shorten the number of lines of code necessary to find the solution. However, the memory and time complexity of the programs isn’t necessarily made efficient by doing this.