Project 3

**THREE RECURSIVE FUNCTIONS**

Full Name: Prashul Shrestha

Section #: 2

Project #: 3

Due Date: February 7, 2017.

**Design** **Document**

**Introduction**

**Recursion** is a programming technique in which a function calls itself. Recursion is another way of implementing repetitive computations of associating loops. It is more efficient than loops but, the time consumption may not vary. In recursion, the function is called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or other instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem. It can be used to find the Palindrome words, check the value present in an array, and so on.

**Data** **Structures**

The program uses two data structures to store and hold the characters of a string (array of characters) called Sentence and another one int array[] to store the prime divisible numbers. The first array is used in the function of Palindrome to store the characters of the inputted sentence in a char array and pass to the function bool Palindrome() to use recursive to check for palindrome. The other array stores the numbers that are divides the inputted prime number and later prints the items in the array.

**Functions**

The program uses three recursive functions to implement the commands entered by the user. Each functions are called while as per the menu-function is implemented. The list of the functions are given below:

1. int GCD(): This function takes the two number given by the user from the main() function

and uses recursive to find the Greatest Common Divisor of the two numbers.

1. bool is\_Palindrome(): This function takes the char array passed by reference from main().

Then the array is checked from front and back characters to check if

the string is palindrome and returns the bool type.

1. int Prime\_fact(): This function takes the number from the main(), checks if the given

number from user is Prime or not and return the factors of that number.

The program uses **string**.**size**() from **string**.h library to get the length of the string. It also uses **.get(ch)** function from the **c++** library to get one character at time to store it in the char array only the alphabets character.

**Menu**() is a Menu-Driven function made using switch case which displays the commands to the user and takes the command from the user to compute GCD, Palindrome and Prime Factorization, respectively. The **main**() calls the function as per the user’s given command from the cases in switch case statement.

**The Main Program**

The main program provides the list of commands to the user. The main program uses the **Switch**-**Case** to take in the command entered by the user and calls the function implemented on the same .cpp file and implement the command. The user can get the Greatest Common Divisor of two numbers, check for a string as Palindrome, and get the factors of Prime number using the recursive technique instead of loop. The command is directly inputted by the user as ‘g’, ‘p’ and ‘f’, respectively to call the function to find **GCD**, check Palindrome and return the prime factors of a number.

User Document

**Recursion** is a programming technique in which a function calls itself. Recursion is another way of implementing repetitive computations of associating loops. It is more efficient than loops but, the time consumption may not vary. In recursion, the function is called to solve a problem, a recursive function passes a smaller version (or versions) of the problem to another instance (or other instances) of itself, then uses the results returned by the recursive call or calls to build a solution to the original problem. It can be used to find the Palindrome words, check the value present in an array, and so on.

The main program named Recursive.cpp can be compiled and run, using the code:

g++ Recursive.cpp

a.out

**g++** function will compile the function and make it ready to be run using **a.out**. The function will prompt the following output:

**Ouptut:**

**Today, we are going to find the GCD, Check Palindrome String and Prime Factorization of a number.**

**Type 'g' to get the GCD of two number.**

**Type 'p' to check if the String is Palindrome.**

**Type 'f' to get Prime Factorization of two numbers.**

**Type 'm' to diplay the menu.**

**Type 'q' to quit the program.**

**Thank you.**

**\*\*\*\*\*-----------\*\*\*\*\***

**Enter the command :**

:- The user will prompt the computer as per their desired command and the program will respond accordingly.

**Summary**

In this project, we implemented **recursive** function in the program to find the Greatest Common Divisor of two numbers, check if the string is Palindrome and find the Prime Factorization of a number. Recursive function is a type of function in which the function calls itself, implementing changes in the arguments each time it calls itself which is coded within the same function.

In this program, the program takes in command from the user in the main function and uses **switch**() case statement to go to another sub function where there are functions that carry out the code as required per the function. First, the user finds the GCD of two numbers, secondly, checks the string for Palindrome and lastly, returns the Prime Factorization of two numbers.

In this program we used simple coding with the implementation of recursion instead of loops so, completing this project, I became through with the use of Recursion in the future coding which can be time efficient and more importantly won’t be much complicated. Since, in this program, it uses **.get(ch)** to get one characters at time from the string inputted and store that taken character into a char **array[].**

Overall, I believe, this program is in the best form, it can be made more effective, by using more sub-functions, like a function to get the string from the user and carry out rest code on that function and call the main **Palindrome()** function from that sub-function. Other way to make it easier was by using the array and the Boolean type as global variable.

I had one doubt while doing this project, that is while using the **.get(ch)** to get one character and store into the char array[]. Since, I didn’t know the length of the string, I had to set limit to space in **array[]** and while setting a **for** loop to store one char at time, I went into endless loop since the length of the string was unknown and the limit of the loop was set off-bound.