**Sandesh Ghimire**

**CSCI 301 section 3**

**Computer Science 2**

**Project 2**

**10th September**

**Design Document**

**Introduction:**

           Recursion is the technique where a function does computation by calling itself. We can do the computation that involves looping through recursion. In recursion, a function calls itself with some or full part of it eventually terminating it by returning when the condition is satisfied. While doing recursion, we should try to solve the smaller version of the problem first, think about solving the larger problem using the smaller version technique, and carefully use the base conditions. The following three programs use recursion to deal with the problem.

The first program simply adds up the digits of any input number. The input integer is sent as a parameter while calling the function (AddUp). The function AddUp applies a base condition to add up the digits eventually returning the sum of digits.

The second program finds out if the entered phrase is palindrome or not. The main function asks to input a phrase to the user. That phrase is sent to the function Palindrome which firstly uses base condition to find the inverse of the filtered phrase. The second condition checks if the phrases are equal or not. Finally, the program goes to postcondition that returns a value terminating the recursion.

The third program finds out the prime factor of an entered number. The input from a user is taken from the main function. After receiving the input, a function PrimeFactor is called with the input integer and value of i as a parameter. This function starts with the base condition which runs the 2nd condition if satisfied. the second condition finds a prime factor, prints it, and recalls the function. With the postcondition, it terminates the recursion printing the last remaining number.

**Data Structure**

           The first program uses one data structure i.e. integer. It is used to get input as well as output. The input integer is taken as a parameter to the function *AddUp*. *AddUp* function then returns the sum which is also an integer.

           The second program uses two data structures i.e. Strings and Boolean. Strings are used to get input and compute the strings to find palindrome. After comparing the input and computed strings a Boolean value is returned to the main function.

           The third program uses one data structure i.e. integer. An integer is used to take the input value and commutate the solution. After finding the result required an integer is printed.

**Functions**

-1st program (ADD UP THE DIGITS OF AN INTEGER)

           It uses one function i.e. AddUp. It has a parameter of an integer data type. The parameter is the input that is taken from the user in the main function. This function uses a base condition checking if the input number is greater than 10 or not. If it is greater than 10 then it adds up the value of the last digit to the sum by calling the function itself. Now, the function is called with the value of Num1 divided by 10 as a parameter. If the value is less then 10, it terminates the recursion by returning the value of Num1.

-2nd program (IS AN INPUT STRING IS A PALINDROME?)

           It uses one function with the name Palindrome. It carries two parameters of string data type. It returns a Boolean value after calculating at last when the recursion terminates. It prints weather the provided phrase is palindrome or not within this function. A base condition is used to find the inverse of the filtered phrase (The original phrase without spaces and uppercase). The 2nd condition check if the Filtered string and inversed string are same or not. Finally, it moves to postcondition terminating the recursion.

-3rd program (PRIME FACTORIZATION)

           It uses only one function with the name PrimeFactor that returns an integer to the main function when it is called. It carries two parameters of integer data type (Num1 and i). The function starts with the base condition that checks if the input integer (Num1) is smaller then the square root of itself. If the base condition is satisfied it goes to 2nd condition that checks if the remainder when Num1 is divided by I is zero or not. If the second condition is satisfied it prints the value of i and then calls the function itself with the parameter (Num1/10 and i). If the second condition is not satisfied, it simply increases the value of i by 1 and then calls the function itself with the same value of Num1 and the incremented value of i. Lastly, the postcondition prints the number that cannot be divided by itself i.e. a prime number, returning the value of i to the main function.

**The Main Program:**

I In the first program, the main function asks for input displaying a message. The number is assigned to a variable called Num. After assigning the value, the function AddUp is called with the variable Num as its parameter.

           In the second program, the main function asks for input of string by displaying a message. That value is stored in a variable named Str. After assigning the value, the phrase is filtered and stored in a variable *FilterStr*. Now, the function Palindrome is called which prints if the value is palindrome or not.

           In the third program, the main function asks for an integer as input by displaying the message. It then calls the function PrimeFactor which displays the calculated factors.

**Code**

**1st program**

//CSCI 301

/\*This program simply adds up every digits of the input number. The input is taken from the main function.

The function AddUp is called with the input integer as parameter. The function AddUp starts with the post condition

which terminates the recursion if satisfied and returns the left-over value. If the base condition is satisfied the function

adds the digit and then stores it to a variable i.e. Sum. Finally, the value of Sum is returned printing the value and then

terminating the program.

\*/

#include <iostream>

using namespace std;

int AddUp(int); //declaring and initializing the function

int main() //the main function

{

int Num; //declaring a variable

cout << "Enter an Integer: "; //displaying massage for input

cin >> Num; //Input and store value to Num

cout << AddUp(Num); //calling the function AddUp with the input number as parameter

return 0;

}

int AddUp(int Num1)

{

//The AddUp function starts

int Sum = Num1 % 10; //taking out the last digit of a Number

if (Num1 < 10)

{

//post condition

return Num1; //return value and terminate the recursion

}

else

{

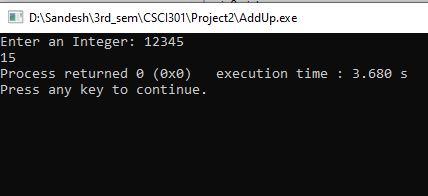
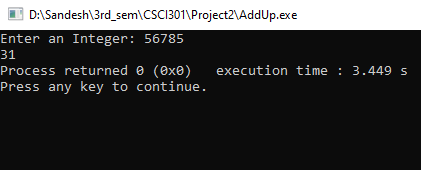
//base condition

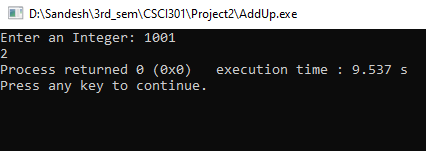
Sum = Sum + AddUp(Num1 / 10); //recall the function

}

return Sum; //return the last value

}





**2nd program**

//CSCI 301

/\*this program checks if the phrase is palindrome or not. firstly, the main function asks input to the users. It uses a loop to

remove spaces and special characters and then

calls the function (Palindrome) with the input string as parameter. In the function (Palindrome) some static value is declared

and then base condition is set. After going through base condition, another condition is applied to

finally check the strings and display the message.

\*/

#include <iostream>

#include <strings.h>

using namespace std;

bool Palindrome(string, string);//declaring the function

int main()//the main function

{

string Str="";//declaring the variable

string FilterStr="";//declaring the variable

string InverseStr;//declaring the variable

cout << "Enter a line that might be a palindrome: "<<endl;//displaying for input

getline(cin, Str);//getting the input

for(int i=0;i<Str.length();i++){//using loop to remove spaces and making every character lowercase.

if((Str[i] >= 'A' && Str[i] <= 'Z') || (Str[i] >= 'a' && Str[i] <= 'z')){

FilterStr.push\_back(tolower(Str[i]));//FilterStr stores the value filtered phrase

}

}

Palindrome(FilterStr,InverseStr);//function is called

return 0;

}

bool Palindrome(string Str1,string InverseStr){//Palindrome function

static int i=Str1.length();//declaring the static value so the integer won't change next time

if (i>0){//base condition precondition

i=i-1;//decreasing value of i by 1

InverseStr.push\_back(Str1[i]);//pushing the last string to first of InverseStr

Palindrome(Str1,InverseStr);//calling function itself

}

else if (Str1==InverseStr){//2nd condition to check palindrome

cout<<" The string is a palindrome."<<endl;//output displayed

return true;//terminating the recursion

}

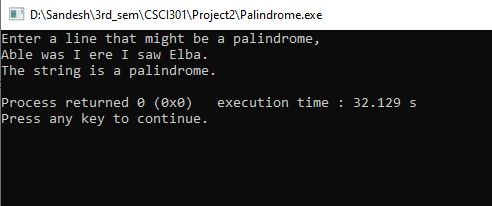
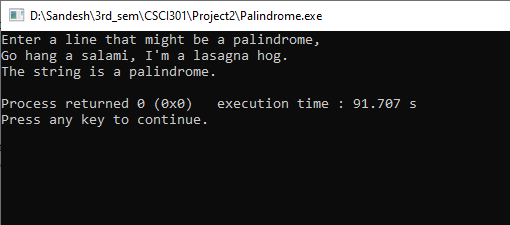
else{//postcondition

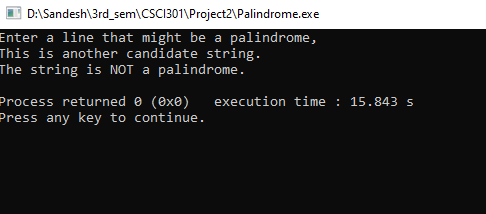
cout<<" The string is NOT a palindrome."<<endl;//output the message

return false;//terminating the recursion

}

}





**3rd program**

//CSCI 301

/\*

This program finds the prime factor of any numbers. The input from a user is taken from the main function.

After receiving the input a function (PrimeFactor) is called with the input integer and value of i as parameter.

This function starts with the base condition which runs the 2nd condition if satisfied. the second condition finds a

prime factor, prints it and recalls the function. With the post condition it terminates the recursion printing the last

remaining number.

\*/

#include <iostream>

#include <cmath> // including cmath library to use sqrt function

using namespace std;

int PrimeFactor(int, int); //declaring the function

//The main function

int main()

{

int Num; //declaring the variable that holds input

cout << "Enter a positive Integer: "; //asking an integer

cin >> Num; //Assigning the input to Num variable

int i = 2; //initializing the value of i

cout << "The prime factors of " << Num << " are: "; //Comment that displays the prime factors

PrimeFactor(Num, i); //calling the PrimeFactor function

return 0;

}

//PrimeFactor function

int PrimeFactor(int Num1, int i)

{

if (i < sqrt(Num1))

{

//starting the base condition

if (Num1 % i == 0)

{

//2nd condition where code looks for prime factor and prints it

cout << i << " "; //printing the factor

PrimeFactor(Num1 / i, i); //recalling the function

}

else

{

i = i + 1; //increasing the value of i

PrimeFactor(Num1, i); //recalling the function

}

}

else

{

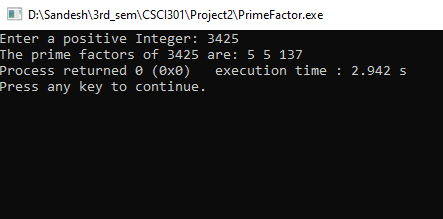
//post condition

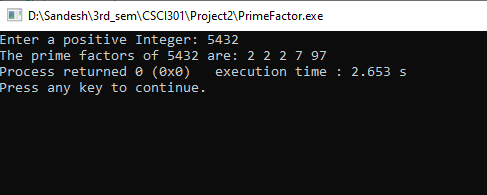
cout << Num1 << " "; //if not negative print the last value of remaining factor

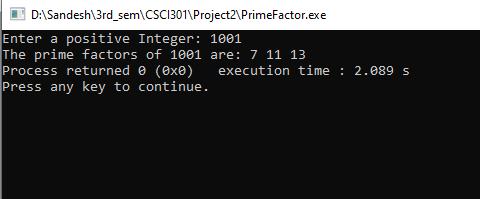
return i; //terminating the recursion

}

}



****



**Summary**

In this project, we learned about recursion. We entered codes to solve three problems using recursion. We used strings as well as integers to find our solutions. In the first and third programs, we used integer data type to add up each digit in a number and find the prime factor using recursion. In the second program, we used string data type to find inverse and compare two strings so that we can tell if the phrase is palindrome or not.

This program lets us workout on how to use recursion instead of a loop to solve the same problem.

I found the second program the most interesting because this program can be done in different ways for example, we can compare each character from the front and behind at the same time to check if they are the same or not. But I first find out the inverse of the phrase and then compared it with the filtered phrase (the original phrase without space, and upper case) as it seems easier to do rather than comparing one by one. We could have made a different recursive function in program 2 to filter (the original phrase without space, and upper case) but instead, I used loop as it is not the core of the problem asked. Other first and second programs were quite easy to do. They also have many different ways to do the same problem. For example: In program 3 we could have set the value of *i*3 at first and then increased the value of the variable by 2. It saves a lot of time.