```
***********************************************
* PROGRAMMED BY : Faris Hijazi
* CLASS         : CS1A
* SECTION       : MW: 7:30P
* Assignment #4 : Recursion Performance
***********************************************

1 - Calculate and Display Factorial of a Number
2 - Calculate and Display Fibonachi Series of a Number
3 - Compare Performance for Factorial Implementations
4 - Compare Performance for Fibonachi Implementations
0 - Exit
enter a command (0 to exit): 1

Enter a number n: 9
calculating...
Factorial of 9 is: 362880

1 - Calculate and Display Factorial of a Number
2 - Calculate and Display Fibonachi Series of a Number
3 - Compare Performance for Factorial Implementations
4 - Compare Performance for Fibonachi Implementations
0 - Exit
enter a command (0 to exit): 2

Enter a number n: 15
Fibonachi series:
0,
1,
1,
2,
3,
5,
8,
13,
21,
34,
55,
89,
144,
233,
377

1 - Calculate and Display Factorial of a Number
2 - Calculate and Display Fibonachi Series of a Number
3 - Compare Performance for Factorial Implementations
4 - Compare Performance for Fibonachi Implementations
0 - Exit
enter a command (0 to exit): 3

Enter a number n: 8
```

```

Measuring exicution time for recursive...
It took the program 7 microseconds to execute.

Measuring execution time for non recursive...
It took the program 3 microseconds to execute.

1 - Calculate and Display Factorial of a Number
2 - Calculate and Display Fibonachi Series of a Number
3 - Compare Performance for Factorial Implementations
4 - Compare Performance for Fibonachi Implementations
0 - Exit
enter a command (0 to exit): 3

Enter a number n: 15

Measuring exicution time for recursive...
It took the program 14 microseconds to execute.

Measuring execution time for non recursive...
It took the program 11 microseconds to execute.

1 - Calculate and Display Factorial of a Number
2 - Calculate and Display Fibonachi Series of a Number
3 - Compare Performance for Factorial Implementations
4 - Compare Performance for Fibonachi Implementations
0 - Exit
enter a command (0 to exit): 4

Enter a number n: 15

Measuring exicution time for recursive...
It took the program 1786 microseconds to execute.

Measuring execution time for non recursive...
It took the program 647 microseconds to execute.

1 - Calculate and Display Factorial of a Number
2 - Calculate and Display Fibonachi Series of a Number
3 - Compare Performance for Factorial Implementations
4 - Compare Performance for Fibonachi Implementations
0 - Exit
enter a command (0 to exit): 4

Enter a number n: 30

Measuring exicution time for recursive...
It took the program 735198 microseconds to execute.

Measuring execution time for non recursive...
It took the program 351 microseconds to execute.
```

```

1 - Calculate and Display Factorial of a Number
2 - Calculate and Display Fibonachi Series of a Number
3 - Compare Performance for Factorial Implementations
4 - Compare Performance for Fibonachi Implementations
0 - Exit
enter a command (0 to exit): 0
```

```
/**************************************************************************
 * AUTHOR     :Faris Hijazi
 * STUDENT ID :1039438
 * LAB #12    :Recursion Performance
 * CLASS      :CS1B
 * SECTION    :MW: 7:30pm
 * DUE DATE   :4/30/19
 **************************************************************************/

#ifndef HEADER_H_
#define HEADER_H_
#include <string>
#include <iostream>
#include <iomanip>
#include <limits>
#include <ios>
#include<chrono>
#include<ctime>
using namespace std::chrono;
using namespace std;

enum menu
{
    EXIT,
    FAC,
    FIB,
    FACP,
    FIBP
};

void PrintHeader(ostream &output,         //output device
                 char exersize,           //lab or assignment?
                 string exersizeName,     //lab or assignment name
                 int num,                 //lab or assignment name
                 string names);           //names of programmer(s)

long long factorial(long long num);       //num to calulate factorial of

long long factorialR(long long num);      //num to calulate factorial of

string fib(long num);                     //numbers in series to display

long fibR(long num);                      //number in series to display

string outputArray(long arr[],            //array to output
                   int num);              //number of elements to output

void outputMenu();

int menuInput();

```

```
#endif /* HEADER_H_ */
```

```cpp
/*************************************************************************
 * AUTHOR     :Faris Hijazi
 * STUDENT ID :1039438
 * LAB #12    :Recursion Performance
 * CLASS      :CS1B
 * SECTION    :MW: 7:30pm
 * DUE DATE   :4/30/19
 *************************************************************************/

#include "header.h"

int main()
{
    int n;                                  //IN - number to calculate fib of factorial
    int i;                                  //CALC - LCV in for loop
    int menuOpt;                            //IN&CALC - menu option user chooses
    int numEx;                              //CALC - LCV in for loop, num of executions
                                            //     - when calculation ex time
    high_resolution_clock::time_point t1;//CALC - time before execution
    high_resolution_clock::time_point t2;//CALC - time after execution
    long long duration1;                    //CALC&OUT - difference between t1 and t2 in
                                            //         - microseconds

    PrintHeader(cout,'A',"Recursion Performance",4,"Faris Hijazi");

    menuOpt = menuInput();

    while(menuOpt != 0)
    {
        switch (menuOpt)
        {
            case EXIT:
                break;

            case FAC:
                cout << endl << "Enter a number n: ";
                cin >> n;
                cout << "calculating...\n";
                cout << "Factorial of " << n << " is: " << factorialR(n) << endl;
                break;

            case FIB:
                cout << endl << "Enter a number n: ";
                cin >> n;
                cout << "Fibonachi series: ";
                for(i=0;i < n; i++)
                {
                    cout << endl << fibR(i);
                    if(i < n-1)
                    {
                        cout << ',';
                    }
                }
                cout << endl;
                break;
            case FACP:
                cout << endl << "Enter a number n: ";
                cin >> n;
                cout << endl;

                cout << "Measuring exicution time for recursive...\n";

                t1 = high_resolution_clock::now();
                for(numEx=0;numEx<=100;numEx++)
                {
                    factorialR(n);
                }
                t2 = high_resolution_clock::now();
                duration1 = duration_cast<microseconds>( t2 -t1 ).count();

                cout << "It took the program "<< duration1 << " microseconds to execute.\n\n";

                cout << "Measuring execution time for non recursive...\n";
                t1 = high_resolution_clock::now();
                for(numEx=0;numEx<=100;numEx++)
                {
                    factorial(n);
                }
                t2 = high_resolution_clock::now();
                duration1 = duration_cast<microseconds>( t2 -t1 ).count();

                cout << "It took the program "<< duration1 << " microseconds to execute.\n";

                break;

            case FIBP:
                cout << endl << "Enter a number n: ";
                cin >> n;
                cout << endl;

                cout << "Measuring exicution time for recursive...\n";

                t1 = high_resolution_clock::now();
                for(numEx=0;numEx<=100;numEx++)
                {
                    for(i=0;i < n; i++)
                    {
                        fibR(i);
                    }
                }
                t2 = high_resolution_clock::now();
                duration1 = duration_cast<microseconds>( t2 - t1 ).count();

                cout << "It took the program "<< duration1 << " microseconds to execute.
\n\n";

                cout << "Measuring execution time for non recursive...\n";
                t1 = high_resolution_clock::now();
                for(numEx=0;numEx<=100;numEx++)
                {
                    fib(n);
                }
                t2 = high_resolution_clock::now();
                duration1 = duration_cast<microseconds>( t2 -t1 ).count();

                cout << "It took the program "<< duration1 << " microseconds to execute.
\n";
                break;

        }
        menuOpt = menuInput();
    }

    return 0;
}
```

```cpp
#include "header.h"
/**************************************************************************
 * This function will find the factorial of an int num
 *------------------------------------------------------------------------
 * INPUT:
 *       num - long long integer
 * OUTPUT:
 *       factorial of num
 *************************************************************************/
long long factorial(long long num)
{
    long long factorial; //OUT - holds factorial of num

    factorial = num;

    if (num <= 1)
    {
       factorial = 1;
    }
    while(num-1 > 0)
    {
        factorial = factorial * (--num);
    }
    return factorial;
}

/**************************************************************************
 * This function will output the fibonachi series, up to (num) numbers
 *------------------------------------------------------------------------
 * INPUT:
 *       num - number of numbers in series to calculate
 * OUTPUT:
 *       fibonachi series
 *************************************************************************/
string fib(long num)
{
    int i;          //CALC - int used in for loop
    long fib;       //CALC - stores result of calculation
                    //       for next number in series
    long series[50];//CALC - array of fib series
    string output;  //OUT  - string of series to output

    if(num <= 1)
    {
        series[0] = 1;
    }
    else
    {
        fib = 0;
        for(i = 0;i < num;i++)
        {
            if(i == 0)
            {
                series[i] = 0;
            }
            else if (i == 1)
            {
                series[i] = 1;
            }
            else
            {
                series[i] = series[i-1] + series[i-2];
            }
        }
    }
    return outputArray(series, num);
}

/******************************************************************************
 * This function will get a menu infut from the user and error check the input
 *-----------------------------------------------------------------------------
 * INPUT:
 *      NA
 * OUTPUT:
 *      menuOpt
 ******************************************************************************/
int menuInput()
{
    int menuOpt;
    bool invalid = false;
    do
    {
        outputMenu();
        if(!(cin >> menuOpt))
        {
            cout << "\n**** Please input a number between 0 and 4 ****\n";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            invalid = true;
        }
        else if(menuOpt < 0 || menuOpt > 4)
        {
            cout << "\n**** The number " <<  menuOpt << " is an invalid entry ****\n";
            cout << "**** Please input a number between 0 and 4 ****\n";
            invalid =  true;
        }
        else
        {
            cin.ignore(1000, '\n');
            invalid = false;
        }
    }while(invalid);

    return menuOpt;
}
/*****************************************************************************
 * This function will output a menu of options for the user to pick from
 *----------------------------------------------------------------------------
 * INPUT:
 *      NA
 * OUTPUT:
 *      NA
 *****************************************************************************/
void outputMenu()
{
    cout << "\n1 - Calculate and Display Factorial of a Number\n";
    cout << "2 - Calculate and Display Fibonachi Series of a Number\n";
    cout << "3 - Compare Performance for Factorial Implementations\n";
    cout << "4 - Compare Performance for Fibonachi Implementations\n";
    cout << "0 - Exit\n";
    cout << "enter a command (0 to exit): ";
}

/*****************************************************************************
 * This function will output a series of numbers from an array
 *----------------------------------------------------------------------------
 * INPUT:
 *      arr[]   - array of numbers to output
 *      num     - number of numbers to output from array
 * OUTPUT:
 *      NA
 *****************************************************************************/
string outputArray(long arr[], int num)
{
    string output;
    int i;
    for(i=0;i<num;i++)
    {
        output += to_string(arr[i]);
        if(i < num-1)
        {
            output += ",";
        }
    }
    return output;
}

/*****************************************************************************
 * This function will output the class header using ostream
 *----------------------------------------------------------------------------
 * INPUT:
 *      output      - output file variable
 *      exersize    - Lab or Assignment
*       exersizeName- name of exersize
*       num          - number of Lab/Assignment
*       names        - names of programmers
* OUTPUT:
*       header
******************************************************************************/
void PrintHeader(ostream &output, char exersize, string exersizeName,  int num, string names)
{

    int colWidth;  //CALC - changes based on exersize
    string asType; //CALC - changes based on exersize

    if(exersize == 'L')
    {
        asType = "Lab";
        colWidth = 9;
    }
    else
    {
        asType = "Assignment";
        colWidth = 2;
    }

    output << left;
    output <<"************************************************\n";
    output <<"* PROGRAMMED BY : "  << names << endl;
    output <<"* "<< setw(14) << "CLASS"   << ": " << "CS1A"    << endl;
    output <<"* "<< setw(14) << "SECTION" << ": " << "MW: 7:30P"  << endl;
    output <<"* "<< asType << " #" << setw(colWidth)  << num   << ": " << exersizeName << endl;
    output <<"************************************************\n";
    output << right;
}


```

```cpp
/**************************************************************************
 * AUTHOR     :Faris Hijazi
 * STUDENT ID :1039438
 * LAB #12    :Recursion Performance
 * CLASS      :CS1B
 * SECTION    :MW: 7:30pm
 * DUE DATE   :4/30/19
 **************************************************************************/

#include "header.h"
/*************************************************************************
 * This function will find the factorial of an int, num recursively
 *-----------------------------------------------------------------------
 * INPUT:
 *      num - long long integer
 * OUTPUT:
 *      factorial of num
 *************************************************************************/
long long factorialR(long long num)
{
    if (num <= 1)
    {
        return 1;
    }
    else
    {
        return num*factorialR(num-1);
    }
}

/*************************************************************************
 * This function will output the fibonachi number at num recursively
 *-----------------------------------------------------------------------
 * INPUT:
 *      num - number of numbers in series to calculate
 * OUTPUT:
 *      fibonachi number
 *************************************************************************/
long fibR(long num)
{
    if (num <= 1)
    {
        return num;
    }
    return fibR(num-1) + fibR(num-2);
}
```