

output.txt

```
1 *****
2 * PROGRAMMED BY : Faris Hijazi
3 * CLASS       : CS1A
4 * SECTION     : MW: 7:30P
5 * Assignment #4 : Recursion Performance
6 *****
7
8 1 - Calculate and Display Factorial of a Number
9 2 - Calculate and Display Fibonacci Series of a Number
10 3 - Compare Performance for Factorial Implementations
11 4 - Compare Performance for Fibonacci Implementations
12 0 - Exit
13 enter a command (0 to exit): 1
14
15 Enter a number n: 9
16 calculating...
17 Factorial of 9 is: 362880
18
19 1 - Calculate and Display Factorial of a Number
20 2 - Calculate and Display Fibonacci Series of a Number
21 3 - Compare Performance for Factorial Implementations
22 4 - Compare Performance for Fibonacci Implementations
23 0 - Exit
24 enter a command (0 to exit): 2
25
26 Enter a number n: 15
27 Fibonacci series:
28 0,
29 1,
30 1,
31 2,
32 3,
33 5,
34 8,
35 13,
36 21,
37 34,
38 55,
39 89,
40 144,
41 233,
42 377
43
44 1 - Calculate and Display Factorial of a Number
45 2 - Calculate and Display Fibonacci Series of a Number
46 3 - Compare Performance for Factorial Implementations
47 4 - Compare Performance for Fibonacci Implementations
48 0 - Exit
49 enter a command (0 to exit): 3
50
51 Enter a number n: 8
```

output.txt

```
52
53 Measuring execution time for recursive...
54 It took the program 7 microseconds to execute.
55
56 Measuring execution time for non recursive...
57 It took the program 3 microseconds to execute.
58
59 1 - Calculate and Display Factorial of a Number
60 2 - Calculate and Display Fibonacci Series of a Number
61 3 - Compare Performance for Factorial Implementations
62 4 - Compare Performance for Fibonacci Implementations
63 0 - Exit
64 enter a command (0 to exit): 3
65
66 Enter a number n: 15
67
68 Measuring execution time for recursive...
69 It took the program 14 microseconds to execute.
70
71 Measuring execution time for non recursive...
72 It took the program 11 microseconds to execute.
73
74 1 - Calculate and Display Factorial of a Number
75 2 - Calculate and Display Fibonacci Series of a Number
76 3 - Compare Performance for Factorial Implementations
77 4 - Compare Performance for Fibonacci Implementations
78 0 - Exit
79 enter a command (0 to exit): 4
80
81 Enter a number n: 15
82
83 Measuring execution time for recursive...
84 It took the program 1786 microseconds to execute.
85
86 Measuring execution time for non recursive...
87 It took the program 647 microseconds to execute.
88
89 1 - Calculate and Display Factorial of a Number
90 2 - Calculate and Display Fibonacci Series of a Number
91 3 - Compare Performance for Factorial Implementations
92 4 - Compare Performance for Fibonacci Implementations
93 0 - Exit
94 enter a command (0 to exit): 4
95
96 Enter a number n: 30
97
98 Measuring execution time for recursive...
99 It took the program 735198 microseconds to execute.
100
101 Measuring execution time for non recursive...
102 It took the program 351 microseconds to execute.
```

output.txt

103

1041 - Calculate and Display Factorial of a Number

1052 - Calculate and Display Fibonacci Series of a Number

1063 - Compare Performance for Factorial Implementations

1074 - Compare Performance for Fibonacci Implementations

1080 - Exit

109 enter a command (0 to exit): 0

## header.h

```
1 /*****
2 * AUTHOR      :Faris Hijazi
3 * STUDENT ID  :1039438
4 * LAB #12     :Recursion Performance
5 * CLASS      :CS1B
6 * SECTION    :MW: 7:30pm
7 * DUE DATE   :4/30/19
8 *****/
9
10 #ifndef HEADER_H_
11 #define HEADER_H_
12 #include <string>
13 #include <iostream>
14 #include <iomanip>
15 #include <limits>
16 #include <ios>
17 #include <chrono>
18 #include <ctime>
19 using namespace std::chrono;
20 using namespace std;
21
22 enum menu
23 {
24     EXIT,
25     FAC,
26     FIB,
27     FACP,
28     FIBP
29 };
30
31 void PrintHeader(ostream &output,           //output device
32                 char exersize,              //lab or assignment?
33                 string exersizeName,        //lab or assignment name
34                 int num,                   //lab or assignment name
35                 string names);             //names of programmer(s)
36
37 long long factorial(long long num);         //num to calulate factorial of
38
39 long long factorialR(long long num);        //num to calulate factorial of
40
41 string fib(long num);                      //numbers in series to display
42
43 long fibR(long num);                      //number in series to display
44
45 string outputArray(long arr[],             //array to output
46                   int num);               //number of elements to output
47
48 void outputMenu();
49
50 int menuInput();
51
```

header.h

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

main.cpp

```
1 /*****
2 * AUTHOR      :Faris Hijazi
3 * STUDENT ID  :1039438
4 * LAB #12     :Recursion Performance
5 * CLASS      :CS1B
6 * SECTION    :MW: 7:30pm
7 * DUE DATE   :4/30/19
8 *****/
9
10 #include "header.h"
11
12 int main()
13 {
14     int n;                //IN - number to calculate fib of factorial
15     int i;                //CALC - LCV in for loop
16     int menuOpt;          //IN&CALC - menu option user chooses
17     int numEx;            //CALC - LCV in for loop, num of executions
18                           //      - when calculation ex time
19     high_resolution_clock::time_point t1; //CALC - time before execution
20     high_resolution_clock::time_point t2; //CALC - time after execution
21     long long duration1;    //CALC&OUT - difference between t1 and t2 in
22                           //      - microseconds
23
24     PrintHeader(cout, 'A', "Recursion Performance", 4, "Faris Hijazi");
25
26     menuOpt = menuInput();
27
28     while(menuOpt != 0)
29     {
30         switch (menuOpt)
31         {
32             case EXIT:
33                 break;
34
35             case FAC:
36                 cout << endl << "Enter a number n: ";
37                 cin >> n;
38                 cout << "calculating...\n";
39                 cout << "Factorial of " << n << " is: " << factorialR(n) << endl;
40                 break;
41
42             case FIB:
43                 cout << endl << "Enter a number n: ";
44                 cin >> n;
45                 cout << "Fibonacci series: ";
46                 for(i=0; i < n; i++)
47                 {
48                     cout << endl << fibR(i);
49                     if(i < n-1)
50                     {
51                         cout << ',';
```

```

52         }
53     }
54     cout << endl;
55     break;
56 case FACP:
57     cout << endl << "Enter a number n: ";
58     cin >> n;
59     cout << endl;
60
61     cout << "Measuring execution time for recursive...\n";
62
63     t1 = high_resolution_clock::now();
64     for(numEx=0; numEx<=100; numEx++)
65     {
66         factorialR(n);
67     }
68     t2 = high_resolution_clock::now();
69     duration1 = duration_cast<microseconds>( t2 -t1 ).count();
70
71     cout << "It took the program "<< duration1 << " microseconds to execute.
72     \n\n";
73
74     cout << "Measuring execution time for non recursive...\n";
75     t1 = high_resolution_clock::now();
76     for(numEx=0; numEx<=100; numEx++)
77     {
78         factorial(n);
79     }
80     t2 = high_resolution_clock::now();
81     duration1 = duration_cast<microseconds>( t2 -t1 ).count();
82
83     cout << "It took the program "<< duration1 << " microseconds to execute.
84     \n";
85
86     break;
87 case FIBP:
88     cout << endl << "Enter a number n: ";
89     cin >> n;
90     cout << endl;
91
92     cout << "Measuring execution time for recursive...\n";
93
94     t1 = high_resolution_clock::now();
95     for(numEx=0; numEx<=100; numEx++)
96     {
97         for(i=0; i < n; i++)
98         {
99             fibR(i);
100         }

```

main.cpp

```
101         t2 = high_resolution_clock::now();
102         duration1 = duration_cast<microseconds>( t2 - t1 ).count();
103
104         cout << "It took the program "<< duration1 << " microseconds to execute.
        \n\n";
105
106         cout << "Measuring execution time for non recursive...\n";
107         t1 = high_resolution_clock::now();
108         for(numEx=0;numEx<=100;numEx++)
109         {
110             fib(n);
111         }
112         t2 = high_resolution_clock::now();
113         duration1 = duration_cast<microseconds>( t2 -t1 ).count();
114
115         cout << "It took the program "<< duration1 << " microseconds to execute.
        \n";
116         break;
117
118     }
119     menuOpt = menuInput();
120 }
121
122 return 0;
123 }
```



normal-functions.cpp

```
1 #include "header.h"
2 /*****
3  * This function will find the factorial of an int num
4  *-----
5  * INPUT:
6  *     num - long long integer
7  * OUTPUT:
8  *     factorial of num
9  *****/
10 long long factorial(long long num)
11 {
12     long long factorial; //OUT - holds factorial of num
13
14     factorial = num;
15
16     if (num <= 1)
17     {
18         factorial = 1;
19     }
20     while(num-1 > 0)
21     {
22         factorial = factorial * (--num);
23     }
24     return factorial;
25 }
26
27 /*****
28  * This function will output the fibonacci series, up to (num) numbers
29  *-----
30  * INPUT:
31  *     num - number of numbers in series to calculate
32  * OUTPUT:
33  *     fibonacci series
34  *****/
35 string fib(long num)
36 {
37     int i;           //CALC - int used in for loop
38     long fib;        //CALC - stores result of calculation
39                     //      for next number in series
40     long series[50]; //CALC - array of fib series
41     string output;   //OUT - string of series to output
42
43     if(num <= 1)
44     {
45         series[0] = 1;
46     }
47     else
48     {
49         fib = 0;
50         for(i = 0; i < num; i++)
51         {
```

```

52         if(i == 0)
53         {
54             series[i] = 0;
55         }
56         else if (i == 1)
57         {
58             series[i] = 1;
59         }
60         else
61         {
62             series[i] = series[i-1] + series[i-2];
63         }
64     }
65 }
66 return outputArray(series, num);
67 }
68
69 /*****
70 * This function will get a menu input from the user and error check the input
71 *-----
72 * INPUT:
73 *     NA
74 * OUTPUT:
75 *     menuOpt
76 *****/
77 int menuInput()
78 {
79     int menuOpt;
80     bool invalid = false;
81     do
82     {
83         outputMenu();
84         if(!(cin >> menuOpt))
85         {
86             cout << "\n**** Please input a number between 0 and 4 ****\n";
87             cin.clear();
88             cin.ignore(numeric_limits<streamsize>::max(), '\n');
89             invalid = true;
90         }
91         else if(menuOpt < 0 || menuOpt > 4)
92         {
93             cout << "\n**** The number " << menuOpt << " is an invalid entry ****\n";
94             cout << "**** Please input a number between 0 and 4 ****\n";
95             invalid = true;
96         }
97         else
98         {
99             cin.ignore(1000, '\n');
100             invalid = false;
101         }
102     }while(invalid);

```

```

103
104     return menuOpt;
105 }
106 /*****
107  * This function will output a menu of options for the user to pick from
108  *-----
109  * INPUT:
110  *      NA
111  * OUTPUT:
112  *      NA
113  *****/
114 void outputMenu()
115 {
116     cout << "\n1 - Calculate and Display Factorial of a Number\n";
117     cout << "2 - Calculate and Display Fibonacci Series of a Number\n";
118     cout << "3 - Compare Performance for Factorial Implementations\n";
119     cout << "4 - Compare Performance for Fibonacci Implementations\n";
120     cout << "0 - Exit\n";
121     cout << "enter a command (0 to exit): ";
122 }
123
124 /*****
125  * This function will output a series of numbers from an array
126  *-----
127  * INPUT:
128  *      arr[]    - array of numbers to output
129  *      num      - number of numbers to output from array
130  * OUTPUT:
131  *      NA
132  *****/
133 string outputArray(long arr[], int num)
134 {
135     string output;
136     int i;
137     for(i=0; i<num; i++)
138     {
139         output += to_string(arr[i]);
140         if(i < num-1)
141         {
142             output += ",";
143         }
144     }
145     return output;
146 }
147
148 /*****
149  * This function will output the class header using ostream
150  *-----
151  * INPUT:
152  *      output    - output file variable
153  *      exersize - Lab or Assignment

```

normal-functions.cpp

```

154 *      exersizeName- name of exersize
155 *      num          - number of Lab/Assignment
156 *      names        - names of programmers
157 * OUTPUT:
158 *      header
159 *****/
160 void PrintHeader(ostream &output, char exersize, string exersizeName, int num, string
names)
161 {
162
163     int colWidth; //CALC - changes based on exersize
164     string asType; //CALC - changes based on exersize
165
166     if(exersize == 'L')
167     {
168         asType = "Lab";
169         colWidth = 9;
170     }
171     else
172     {
173         asType = "Assignment";
174         colWidth = 2;
175     }
176
177     output << left;
178     output << "*****\n";
179     output << "* PROGRAMMED BY : " << names << endl;
180     output << "* " << setw(14) << "CLASS" << ": " << "CS1A" << endl;
181     output << "* " << setw(14) << "SECTION" << ": " << "MW: 7:30P" << endl;
182     output << "* " << asType << " #" << setw(colWidth) << num << ": " << exersizeName
<< endl;
183     output << "*****\n";
184     output << right;
185 }
186
187

```

recursive-fuctions.cpp

```
1 /*****
2  * AUTHOR      :Faris Hijazi
3  * STUDENT ID  :1039438
4  * LAB #12     :Recursion Performance
5  * CLASS       :CS1B
6  * SECTION     :MW: 7:30pm
7  * DUE DATE    :4/30/19
8  *****/
9
10 #include "header.h"
11 /*****
12  * This function will find the factorial of an int, num recursively
13  *-----
14  * INPUT:
15  *     num - long long integer
16  * OUTPUT:
17  *     factorial of num
18  *****/
19 long long factorialR(long long num)
20 {
21     if (num <= 1)
22     {
23         return 1;
24     }
25     else
26     {
27         return num*factorialR(num-1);
28     }
29 }
30
31 /*****
32  * This function will output the fibonacci number at num recursively
33  *-----
34  * INPUT:
35  *     num - number of numbers in series to calculate
36  * OUTPUT:
37  *     fibonacci number
38  *****/
39 long fibR(long num)
40 {
41     if (num <= 1)
42     {
43         return num;
44     }
45     return fibR(num-1) + fibR(num-2);
46 }
```