Practice Lab (Week 4): Data Structures

File handling	Q1	Q2	Q3	Total
	10	10	10	30

1. This practice lab covers the following **one week** of lectures in the Syllabus. You are expected to watch these videos prior to doing this lab.

Lecture	Topics	Reading
7	**File Handling:** Using arguments of the main function, reading and writing files.	1. Using arguments of the main function: https://youtu.be/1QzY-yMFHC8 2. Reading and Writing files: https://youtu.be/15bOZuCV4i0
8		

This Lab contains three parts, each part having 10 marks.

Part 1) Evaluating 'Execution time':

The library <Time.h> has three important elements:

1. clock_t: this is our data-type,
2. CLOCKS_PER_SEC: a reserved word which evaluates the number of clocks the timer makes per second. This variable changes for different processors. Faster processors have higher clocks per second
3. clock(): function that measures the number of clocks at a particular instance. It returns the number of clock ticks elapsed since the program was launched. To get the number of seconds used by the CPU, you will need to divide by CLOCKS_PER_SEC.

Write this following program and compile it. If there are any errors, then fix them yourself 😉:

```
#include <stdio.h>
#include <time.h>

void fun()
{
   printf("fun() starts\n");
   printf("Press enter to stop fun \n");
   while(1)
   {
      if ( getchar() )
         break;
   }
   printf("fun() ends \n");
}
```

```
// The main program calls fun() and measures time taken by fun()

int main()
{
   // Calculate the time taken by fun()
   clock_t t1, t2;
   t1 = clock();
   fun();
   t2 = clock() - t1;
   double time_taken = ( (double) t2) /CLOCKS_PER_SEC; // in seconds

   printf("fun() took %f seconds to execute \n", time_taken);
   return 0;
}
```

Part 2) Feeding parameters directly from the console:

There are two extremely important arguments of the main function, namely, argc and argv[]. Here 'argc' stands for argument count and 'argv' stands for argument values. These are variables passed to the main function when it starts executing. When we run a program we can give arguments to that program like:

```
C:\Users\MyComputer\Downloads> Hello.exe 7
```

Here the argument 7 is directly provided in the console and the program will print Hello a total of seven times. Consider the code mentioned below. Notice, now we are evaluating the execution time by default. Evaluating the execution time should appear in all your Data-Structure codes. Write the code below and save it as "Hello.c"

```
#include <iostream>
#include <bits/stdc++.h>
#include <string>
#include <ctime>
#include <stdio.h>
#include <time.h>

using namespace std;
int main(int argc, char *argv[])
{
     if (argc != 2)
     {
               cout << "This program has 2 arguments:" << endl;
               cout << "[1] Name of the program: Hello.exe” << endl;
               cout << "[2] Parameter - number of times you want to print Hello” << endl;
               cout << endl << endl;
               cout << “For e.g.: typing the following:” << endl
               cout << “Hello.exe 7” << endl
               cout << “Will print Hello seven times” << endl;
     }
```

```
        else
        {
                cout << "Let's start the program" << end

                // Convert string number to integer number
                int length = strlen( argv[1] );
                int tau = 0;
                int p, o = 1 ;
                for( p = len-1 ; p > -1 ; p = p - 1 )
                {
                                tau = tau + ( (int) argv[2][p] - 48 ) * o ;
                                o = o * 10 ;
                }

                // Printing hello tau number of time
                for ( int i = 1; i <= tau; i++)
                        cout << "[" << i << "]" << "  Hello" << endl;
        } // end of else
        return 0;
    } // end of main function
```

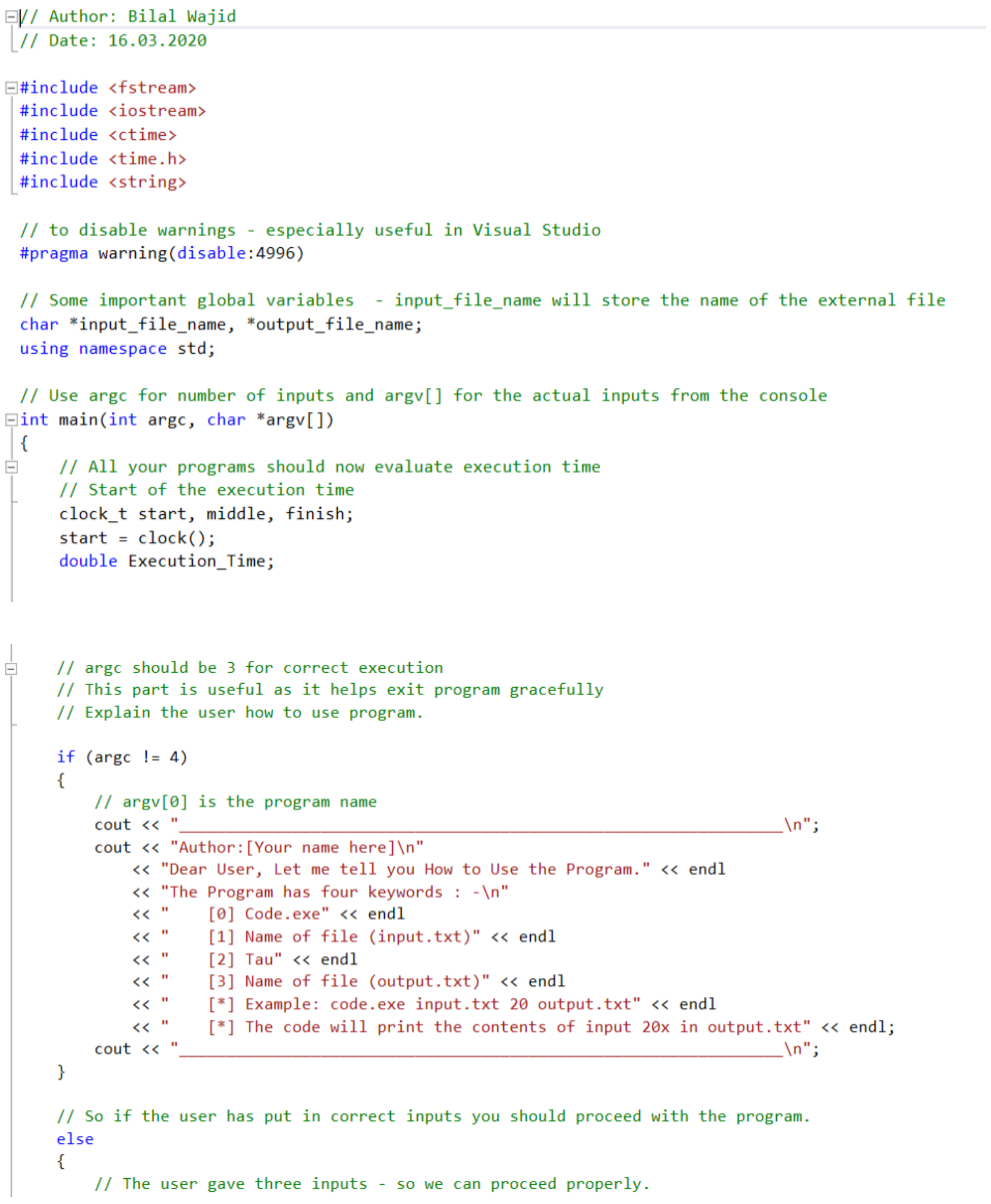
Save the above program as "Hello.c". Compile it on the console and run it directly. See how the output looks like.

Part 3) Input/Console application:

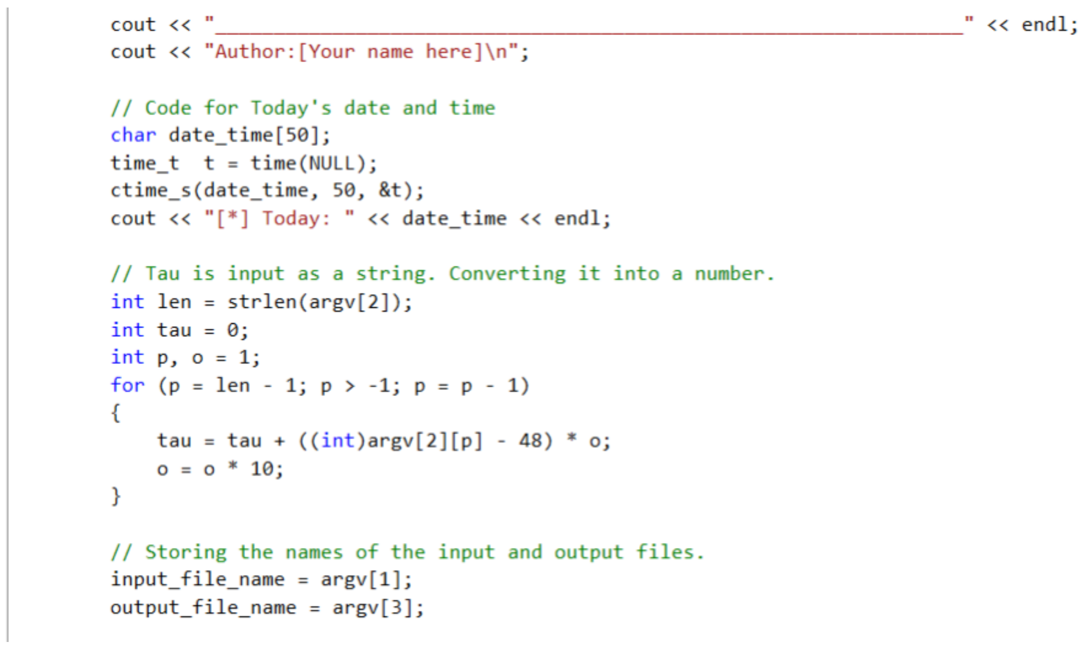
DST teaches you how to handle data using programming structures. The data encompassed within these structures is huge and goes way beyond asking the user to input details using the keyboard. Hence, reading input from an external file is fundamental. This lab teaches how to read an external file while storing the contents in another file.

Focus on the comments, they are mentioned specifically for your understanding. As students are very keen in simply copy-pasting code, the following presents snap-shots (images of the running code) so that you write these codes yourself.

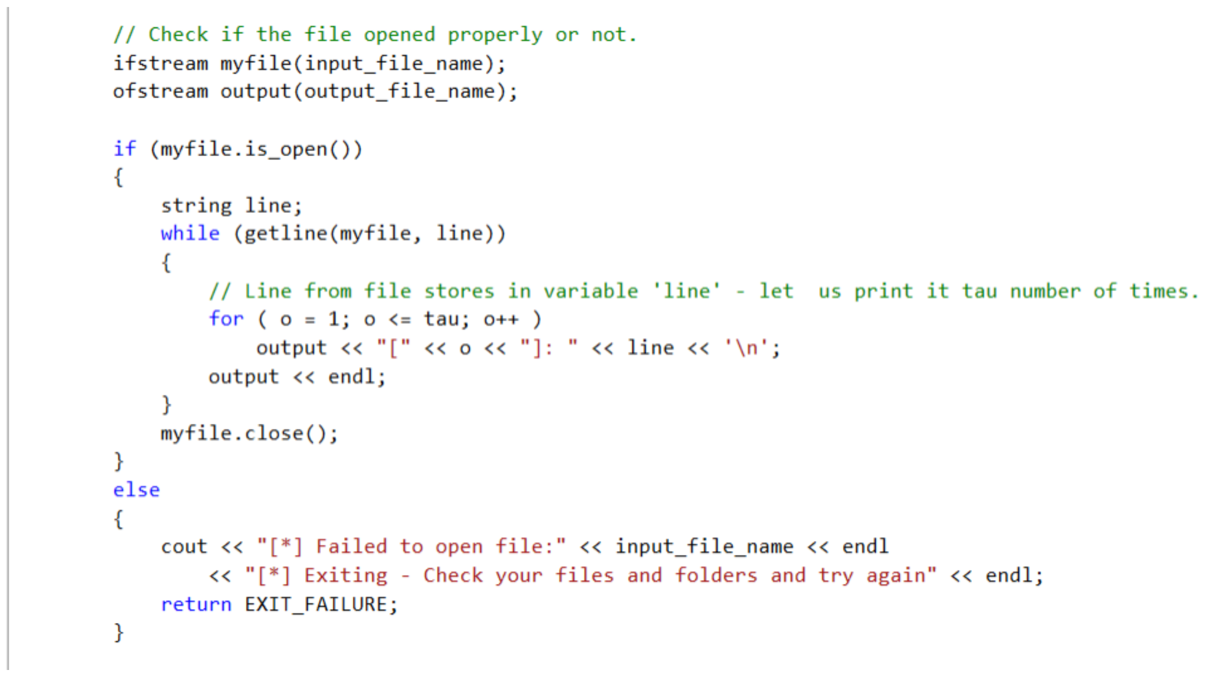
Code:



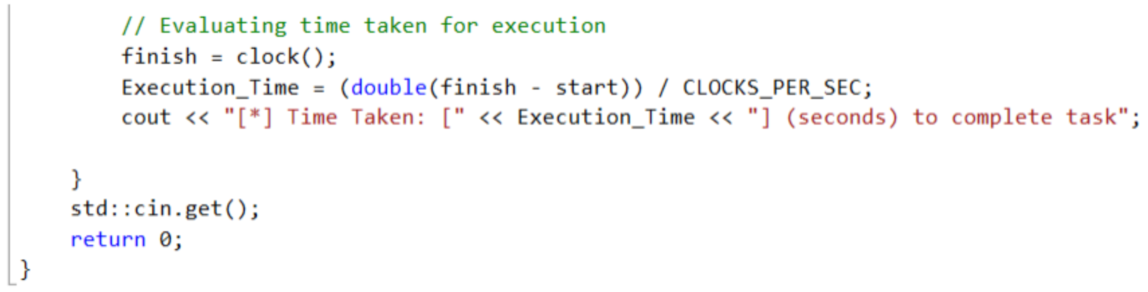
```cpp
// Author: Bilal Wajid
// Date: 16.03.2020

#include <fstream>
#include <iostream>
#include <ctime>
#include <time.h>
#include <string>

// to disable warnings - especially useful in Visual Studio
#pragma warning(disable:4996)

// Some important global variables  - input_file_name will store the name of the external file
char *input_file_name, *output_file_name;
using namespace std;

// Use argc for number of inputs and argv[] for the actual inputs from the console
int main(int argc, char *argv[])
{
    // All your programs should now evaluate execution time
    // Start of the execution time
    clock_t start, middle, finish;
    start = clock();
    double Execution_Time;



    // argc should be 3 for correct execution
    // This part is useful as it helps exit program gracefully
    // Explain the user how to use program.

    if (argc != 4)
    {
        // argv[0] is the program name
        cout << "_________________________________________________________________\n";
        cout << "Author:[Your name here]\n"
            << "Dear User, Let me tell you How to Use the Program." << endl
            << "The Program has four keywords : -\n"
            << "    [0] Code.exe" << endl
            << "    [1] Name of file (input.txt)" << endl
            << "    [2] Tau" << endl
            << "    [3] Name of file (output.txt)" << endl
            << "    [*] Example: code.exe input.txt 20 output.txt" << endl
            << "    [*] The code will print the contents of input 20x in output.txt" << endl;
        cout << "_________________________________________________________________\n";
    }

    // So if the user has put in correct inputs you should proceed with the program.
    else
    {
        // The user gave three inputs - so we can proceed properly.
```



```
        cout << "________________________________________________________________" << endl;
        cout << "Author:[Your name here]\n";

        // Code for Today's date and time
        char date_time[50];
        time_t  t = time(NULL);
        ctime_s(date_time, 50, &t);
        cout << "[*] Today: " << date_time << endl;

        // Tau is input as a string. Converting it into a number.
        int len = strlen(argv[2]);
        int tau = 0;
        int p, o = 1;
        for (p = len - 1; p > -1; p = p - 1)
        {
            tau = tau + ((int)argv[2][p] - 48) * o;
            o = o * 10;
        }

        // Storing the names of the input and output files.
        input_file_name = argv[1];
        output_file_name = argv[3];
```



```
        // Check if the file opened properly or not.
        ifstream myfile(input_file_name);
        ofstream output(output_file_name);

        if (myfile.is_open())
        {
            string line;
            while (getline(myfile, line))
            {
                // Line from file stores in variable 'line' - let  us print it tau number of times.
                for ( o = 1; o <= tau; o++ )
                    output << "[" << o << "]: " << line << '\n';
                output << endl;
            }
            myfile.close();
        }
        else
        {
            cout << "[*] Failed to open file:" << input_file_name << endl
                << "[*] Exiting - Check your files and folders and try again" << endl;
            return EXIT_FAILURE;
        }
```



```
        // Evaluating time taken for execution
        finish = clock();
        Execution_Time = (double(finish - start)) / CLOCKS_PER_SEC;
        cout << "[*] Time Taken: [" << Execution_Time << "] (seconds) to complete task";

    }
    std::cin.get();
    return 0;
}
```

Output:

You are provided a file titled ‘input.txt’ that contains random words. Make sure this file is present in the same folder as the executable version of this program. Type in the code and see how the code works. Run the program in the console.

Rubrics (Associated Marks)

S. No.	Content	Meets Criteria (1)	Marks	Does not meet expectations (0)	Marks
1	Indentation	Perfect	100%	Code not indented properly	0
2	Code works	Code compiles and executes properly for any variable sized matrices	100%	Code has errors	Based on the code
3	Comments	Code is properly commented	100%	Code is not properly commented.	0