Lab13

1. Lab Topics

This lab primarily covers 2D arrays and multiple files.

2. Matrix Addition

There are two files containing different matrices, perform an addition operation on them and export the result into a new file.

Functions to write.

1. Write a function that reads the matrices from the files "matrix1.txt" and "matrix2.txt"
2. Write a function that performs that addition of the two matrices.
3. Write a function that writes the resulting matrix to an output file named "result_matrix.txt".

In the int main

1. Call the function that reads from a file for both the first matrix and the second.
2. Call the function to add these two matrices.
3. Call the function to write the resulting matrix to a file.

This program should demonstrate your ability to:

- Read data from files.
- Manipulate 2D arrays/matrices.
- Perform matrix addition.
- Write the resulting matrix to a file.

3. How to get full marks

To get a 100% on this lab, follow this requirement:

1. Function Documentation and commenting of your program is extremely important (-20 marks).
2. Use multiple files in your program (-30 marks). Create a new folder to save the following file: make sure that only **THE REQUIRED** .cpp file is saved in this folder of yours.
    a. Use the header file (.h) to save your function prototypes and documentations,
    b. Use the header file (.hpp) to save your templated code,
    c. Use the main file (.cpp) to save your main function,
    d. Use the implementation file (.cpp) to save the implementation of your functions.
3. Follow the instructions as stated meeting all requirements for this lab (-50 marks).
4. Make sure your code compile and produce the necessary output before submitting.

5. Push your most recent code in git and submit through canvas as well. The canvas submission should include following files:
    a. The cpp file downloaded from git.
    b. **At least 1 Image file which is the result you got from the addition.**

Others

6. Use good variable names such that one can easily understand a variable's purpose just by looking at the name.
7. The program needs to be intuitive (e.g., display proper messages while you are taking user input or printing the result)
8. Follow all good coding conventions such as proper indentation.
9. Adhere to all coding standards outlined in lab2.
10. Follow the instructions of cloning, making dir, and submitting your code to git as previously discussed in lab01 and lab02.