Final Project Progress Report:

Due date and End date: 10/16/2023, 11:59 PM CDT

Grading Rubrics:

The grading will be as follows:

• Report	**[5 Points]**
• Code	**[5 Points]**

Note:

- Failure to follow standard programming practices will lead to points deduction, even if the program is running correctly. Some of the common places where you could lose points are:
  - Program not compiling successfully: -1 point
    - The TA will run the program in CSX machine.
  - No comments on code (excluding function): -0.25 points
  - No comments on function: -0.25 points
  - Not writing meaningful identifiers: -0.25 points
  - Failure to submit files as specified: -1 point
  - Not appropriate message displayed for user interaction: -1 point

Submission Guidelines:

- Only one group member needs to submit the progress report in the Canvas. Once the progress report is submitted in the Canvas, the member who has submitted the progress report needs to email the TA, cc'ing the instructor and all the group members, about the submission. If any group member has dispute about the submission, it should be brought into the instructor's attention within 24 hrs.
  - Each group is responsible to select the member to submit the file.
  - Failure to submit the files on time will result in the late penalty, as specified in the syllabus, for the entire group.

- Please remember that the due date and the end date are same for the progress report. Once the due/end date is passed, the link will be disabled and no submission will be allowed.

- Progress report submission instruction:
  - Each group (not each group member) needs to submit:
    - A zipped file that includes:
      - pdf report
      - Source code with appropriate extension.
      - readMe.txt file
    - All source code should be submitted in the .pdf format. Please copy paste in textual form. Do not include any screenshot of the code in this pdf. This pdf is used for plagiarism purpose. It should not be inside the zipped file.

The pdf report should highlight:

1. Work distribution of each member in the form of UML. **[3 Marks]**
    - It needs to be specific i.e. it should include:
        - Disintegration of the complex task into smaller and less complex tasks
            - Present the UML diagram to illustrate the program flow
        - Planning and distribution of work
            - In the UML diagram, use color code to represent who is responsible for which part of the project.

2. Each member in the group needs to clearly specify what has been done so far in the project, based on what has been assigned and shown in the UML. **[1 Marks]**
    - All the tested output should be specified. This part should include the validation of work done by each student.
    - All the output should be specified in the report along with the explanation.

3. Each group member needs to clearly specify what is left in the design, based on what has been assigned and shown in the UML. **[1 Marks]**

The code completed so far by each member. **[5 Marks]**

1. Each member has to clearly specify his/her portion of the code. For this, each student's work needs to be in a separate source file.
    - E.g.: If the group chose to use C as a programming language and there are 3 members in the group, then there should be at least three .c files (driver file, i.e. file containing main function, is not considered).
    - each .c file represents one of the group members.
    - You can have as many header (.h) files you want.
2. Whatever work has been done so far need to be compiled from the driver file. The TAwill not run individual member's .c file.

The readMe.txt file needs to include:

1. how to run your program on CSX server,
    - Please include all the details to run your program, without assuming TA knows it.
2. on which CSX server the program is tested.

Illustration of UML:

Example Problem: Read the two integers from the files and perform various arithmetic operations lifeaddition, subtraction, multiplication and division.

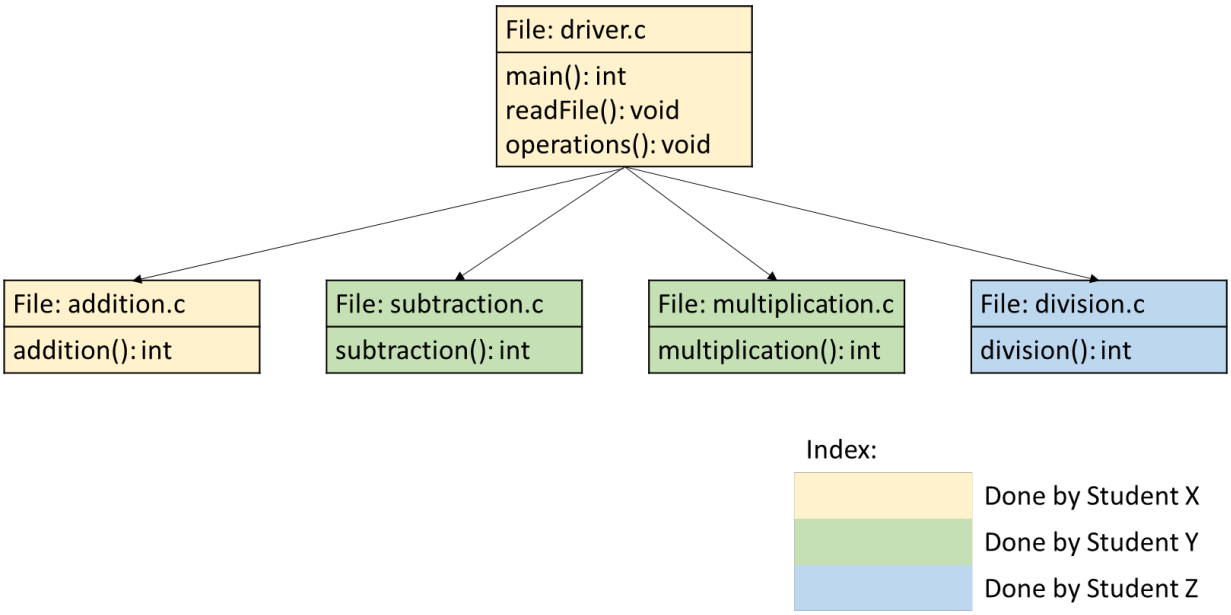

Fig. Illustration of UML for the given program

I believe that the UML is quite self-explanatory. But for the sake of completion, I will explain very briefly:

- 5 files: driver.c, addition.c subtraction.c multiplication.c and division.c
- Driver.c file has 3 functions:
    - main() has a return type of "int". It contains reading input file name from use which contains the data for various arithmetic operation. These data are passed to another function called readFile().
    - readFile() which has a return type void. It reads the data from the file and passes these data another function called operations().
    - operations() function has a return type void and calls all functions defined in another files: addition() from addition.c file, subtraction() from subtraction.c and so on.
- You got the idea for another files.
- There is a color coding used to explain who has done which part:
    - Student X has written code for driver.c and addition.c
    - Student Y has written code for subtraction.c and multipication.c
    - Student Z has written code for division.c