SENG1000 – C/C++ PROGRAMMING

FOCUSED ASSIGNMENT 7 - INTRO TO FILE I/O

OVERVIEW

Write a program to give you practice at creating text and binary files.

GENERAL COURSE OBJECTIVES ADDRESSED IN THIS ASSIGNMENT

- Use file I/O.

ACADEMIC INTEGRITY AND LATE PENALTIES

- Link to Academic Integrity Information
- Link to Late Policy

EVALUATION

- The evaluation of this assignment will be done as detailed in the Marking lecture from Week 2.

PREPARATION

- View Week 10 File I/O videos.
- Review Week 10 File I/O examples.

REQUIREMENTS

Text File

- Write code to create a text file. Its name must be myTextFile.txt. It must contain two lines of content **exactly**:

```
This is line 1.
This is line 2.
```

    - Both lines must end with the \n character.

Binary File

- Write code to create a binary file. Its name must be myBinaryFile.data. It must contain the contents of the following array:

```
const unsigned short kBinaryData[] = { 26946, 24942, 31090,
25632, 29793, 8289, 28518, 8306, 28537, 33141, 39308 };
```

Other Requirements:

- Both output files must be created in the current directory.
- You will have to use the #pragma that you used for getNum(), due to the compiler not liking fopen().
- You must use C-style text file I/O functions as mentioned in the File I/O lecture to create the text file.
- You must use C-style binary file I/O functions as mentioned in the File I/O lecture to create the binary file.
- You must use error checking on all file I/O functions. If there is an error associated with any of the functions, display an appropriate error message indicating which function call failed and quit the program.
- Other than error messages, there is **no** user input or output in this assignment. Having either will result in a mark of 0.
    - This, of course, doesn't include the lines that Visual Studio displays at the end of your program execution, since you have no control over that.
- All other usual course requirements (e.g. file header comment, initialize variables when declared) apply.

CHECKLIST REQUIREMENTS

- Create a requirements checklist. This should contain the specific requirements from this assignment as well as any relevant requirements that have been covered in lecture or that are found in the SET Coding Standards or SET Submission Standards. Do it in whatever form you wish. Hand in your completed checklist in PDF form as checklist.pdf. Not having this checklist will result in a cap of 80 on your mark.

GIT REQUIREMENTS

- You must use GitHub under GitHub Classroom for revision control of your source file. The details were mentioned in an eConestoga announcement earlier. If you have not linked to your account by clicking on the Focused 7 link, please do so now.
- Make git commits every time that you have something working or otherwise substantial.
- It is expected that you will make **regular** git commits with **meaningful** commit comments (describing the changes that you successfully made since the last commit).
    - On an assignment of this size, I would expect at least 3 distinct and meaningful commits. They should be done as you work on your code, so that they are separated by a reasonable amount of time (e.g. it would not be likely that you would complete this assignment in 10 minutes).
    - Take this requirement seriously.
- Failing to use GitHub Classroom will cap your mark at 60.

FILE NAMING REQUIREMENTS

- You must call your source file f7.cpp.
- You must call your checklist checklist.pdf.

SUBMISSION REQUIREMENTS

- Do not hand in any other source files besides those mentioned in the File Naming Requirements.
- Follow the instructions in the SET Submission Standards and the lecture on Submitting Assignments to submit your program. Submit both files to the correct Assignment folder.
- Once you have submitted your files, make sure that you've received the eConestoga e-mail confirming your submission. Do not submit that e-mail (simply keep it for your own records until you get your mark).