Lab3

Due Sep 9, 2020 by 11:59pm **Points** 40 **Submitting** a website url

Available until Sep 10, 2020 at 12:05am

This assignment was locked Sep 10, 2020 at 12:05am.

Lab3:

Submission should be done through the CODE plugin (Upload via CODE)

DUE: Wednesday, September 9th @ 11:59 PM

Background: This lab requires performing some computation after reading from a file and writing the results to a file. The files should be:

1. Path to input file is specified as first command-line argument (i.e., argv[1])
2. Path to output file is specified as second command-line argument (i.e., argv[2])
3. Use std::ifstream and std::ofstream to read and write text file(s)

Procedure: Modify the supplied starter code **lab3.cpp (https://miamioh.instructure.com/courses/129645/files/17445388/download?wrap=1) (https://miamioh.instructure.com/courses/129645/files/17445388/download?download_frd=1)** using the following steps:

1. Add the necessary variables to main function to enable use of command-line arguments **[2 points]**
2. Add an if-check to ensure that the user has specified exactly 2 command-line arguments (i.e., properly check *argc*'s count fo both the input and output file paths). **[4 points]**
   - If not, report the following error: "Specify input and output files" and return 0
3. Open input and output file streams using command-line arguments argv[1] and argv[2] **[4 points]**
   - Ensure the streams were created successfully using the good method
     - Example: *myFile.good()* returns true if the stream is created successfully
   - If the streams are not good, report an error "Error opening input or output streams" and return from main with value of 0
   - NOTE: You must name your output file stream **output** to ensure your code will compile
5. Determine the proper data structure and properly declare/initialize it. **[3 points]**
6. Properly process the input file store the contents of it into that data structure from step 5. **[5 points]**

7. Modify the methods to functions to take in some parameter and perform a computation **[3 points] for each method**
8. Write the results of the computations to an output file with the specified output **[8 points]**:

Sum: [sum]
Min: [min]
Average: [average]
Max: [max]

- **NOTE:** the brackets [] represent the actual numerical value (i.e., it should print something like "Sum: 31"")

The code must pass the style guidelines **[2 points]**

NOTE: typically, we would not return 0 for the failure case as this suggested a successful run. However, the CODE plugin will *correctly* consider it a failed test if there is a return larger then 0 (this would be a failed execution). Since the goal is to *assess* error handling, this is not really a traditional test failure and so we are returning 0.

Input File for Testing:

The file **lab3_input.txt (https://miamioh.instructure.com/courses/129645/files/17056116/download?wrap=1) (https://miamioh.instructure.com/courses/129645/files/17056116/download?download_frd=1)** contains a sample input to test your program.

Functional Testing in the Terminal: Functional testing is the process of checking to ensure that a program is operating correctly – i.e., generates the expected output for a given input. Functional testing is accomplished by comparing the output from a program against a known or expected output. The CODE plugin performs this for you when you submit. However, we can also do this testing directly in the terminal with the following steps:

1. Download the **lab3_expected_output.txt (https://miamioh.instructure.com/courses/129645/files/17445389/download?wrap=1) (https://miamioh.instructure.com/courses/129645/files/17445389/download?download_frd=1)** file and move it to the directory where your lab3 program is
2. Open the terminal and navigate to your lab3 program if you are not there already
3. Run your program with the sample input file **lab3_input.txt (https://miamioh.instructure.com/courses/129645/files/17056116/download?wrap=1) (https://miamioh.instructure.com/courses/129645/files/17056116/download?download_frd=1)** and writing output to a file lab3_output.txt with the following command:
    - ./lab3 lab3_input.txt lab3_output.txt
4. Now, we can compare our results with the expected results using diff (this is the same tool used in the CODE plugin):

- diff lab3_output.txt lab3_expected_output.txt

5. If your program is operating correctly, then the above diff command will generate zero differences – i.e., it will produce absolutely **no output**

Lab 3 Rubric

Criteria	Ratings		Pts
Add the necessary variables to main function to enable use of command-line arguments	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
Add an if-check to ensure that the user has specified exactly 2 command-line arguments (i.e., properly check argc's count fo both the input and output file paths).	**4 pts** **Full Marks**	**0 pts** **No Marks**	4 pts
Open input and output file streams using command-line arguments argv[1] and argv[2]	**4 pts** **Full Marks**	**0 pts** **No Marks**	4 pts
Determine the proper data structure and properly declare/initialize it.	**3 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
Properly process the input file store the contents of it into that data structure from step 5.	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Modify the methods to functions to take in some parameter and perform a computation for each method	**12 pts** **Full Marks**	**0 pts** **No Marks**	12 pts
Write the results of the computations to an output file with the specified output	**8 pts** **Full Marks**	**0 pts** **No Marks**	8 pts
Style guidelines	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
			Total Points: 40