Project 11 (Extra Credit 30 points): Simple Calculator

Possible points are: 30, 25, 20, 15, 10 or 0

Submit Your Solution Using Canvas by 10 PM on Sunday April 23, 2023

To view the order and style of the input and output information for the project, run the provided solution by **typing the following at a terminal window prompt.** (**Note**: Input files must be present in the directory that the terminal window is currently in.)

Sample Solution – for determining vertical spacing differences

/home/work/cpe211/Executables/Project_11/Project_11_solution

Provided sample input files are zipped in the file P11_in.zip

Comparison Script – for determining line differences

/home/work/cpe211data/Project_11/CompareSolution.bash Project_11.cpp

<Project 11 Restrictions>

On Project 11, you may only use any concepts presented in Chapters 1 - 11 of your textbook.
You are not allowed to use global variables. If necessary, global constants may be used. Global constants and variables are declared above the int main() line in a program.

<Starting Project 11>

- Open a terminal window and move (cd) into the **Project_11** directory created in the **CPE211_SPR23** directory (This is the directory structure created in project 1.) The command for this is: **cd CPE211_SPR23/Project_11 (typing cd ~/CPE211_SPR23/Project_11 works as well)**. You will need to modify the names as necessary to match your capitalization style for the two directories.

- Download all needed files from Canvas into this directory. Once you have finished with the program and it compiles without syntax errors, run the executable and verify that the output for your program matches the output from the provided solution executable

- **Once you are satisfied with your solution, submit Project_11.cpp via Canvas**.

NOTE: make sure that you do not change the order in which the information is entered. An automatic script is used to process all lab submissions, and if the order of the input information is modified, the script will not work properly with your program.

NOTE: Output of your program is to match the output of the sample solution.
This match includes all information written to the terminal by the program

<Project 11 Description>

You will write a program that performs the following tasks for a simple mathematical calculator:

(1) Open a user-specified file for input. Prompt for the name of the file, read it into a string variable, echo print it to the terminal and then open it. If the file is not opened, enter into a loop that prints out an error message, resets the input file stream variable **(see the hints section),** obtains a new file name and tries to open the new file. The loop continues until the user successfully enters a valid file name or presses ctrl-c to exit

(2) If the input file is empty, your program should indicate that the input file was empty as shown by the sample solution – note that an empty input file message only is printed out for an empty input file. **(see the hints section on how to perform this test)**

(3) The input file contains several lines of information for processing. There are two types of lines: one for **Math** calculations and one for **Trig** calculations.
   a. **Math Calculation Line Content**: **Type, Operator, Value1, Value2**
      i. Type is **Math**
      ii. Operator is +, -, *, / or % (for add, subtract, multiply, divide or modulo)
      iii. Value1 is the first (left) operand for the operator
      iv. Value2 is the second(right) operand for the operator
   b. **Trig Calculation Line Content**: **Type, Operation, Degrees or Radians indicator, Angle Value**
      i. Type is **Trig**
      ii. Operation is s, c or t (for sine, cosine or tangent)
      iii. Degrees or Radians indicator is d or r
      iv. Angle Value Is the angle in the same units provided by the Degrees or Radians indicator
      v. For Trig calculations use 3.14159265 for the value of pi (store in a double).
      vi. Remember that the trig functions expect the angle in radians not degrees
   c. Look at the provided input files for further understanding of the expected input.

(4) All values are **double** values except for the values used with the modulo operation which requires both operands be integers.

(5) Possible input file errors are:
   a. Invalid calculation type – calculation type is not **Math** or **Trig**.
   b. Invalid operator for Math
   c. Invalid operation for Trig

$\left.\begin{array}{l}\text{b. Invalid operator for Math}\\ \text{c. Invalid operation for Trig}\end{array}\right\}$ $\boxed{\text{two different error messages}}$

(6) Output formatting uses the default output configuration 🡐

(7) Output lines look like the following:
   a. Add: 4+6 = 10
   b. sin(degrees): sin(45) = 0.707107
   c. Identifiers for output for **Math** calculations are: Add, Sub, Mul, Div, Mod
   d. Identifiers for output for **Trig** calculations are: sin, cos, tan

(8) Run the sample solution for the provided input file to see the format of the output

(9) All output is to the terminal. Output of your program is to match that of the sample solution

(10) Functions can be used in this program, although it is not a requirement.

(11) This program can easily be completed using loops and if statements (or switch statement)

<Project 11 C++ Hints>

1. Resetting the input stream variable: In Dev C++ and g++, the clear function must be used on a file stream that is repeatedly reopened in a loop. Therefore use the following statement in your while loop that executes until a valid filename is entered:

   Input_file_stream_var.clear();

   Where **Input_file_stream_var** is the name of your input file stream variable

2. Testing for an empty input file is performed by using a priming read, and then testing the status of the file stream after the priming read – before the while loop is entered. In this case, if the file stream is in the fail state after the priming read, print out an appropriate message and terminate the program. Otherwise proceed to the loop to process and read the rest of the input file

3. Use a while loop to read and process all information in the input file (or a do-while loop can be used)

4. Each line in the input file will contain a calculation type, a type of operation identifier and two values to use with that operation. **All values are separated by spaces**
   a. Calculation type will be a single word of: **Math or Trig**
   b. Operator will be a single character (+, -, *, /, %, s, c or t)
   c. For Math, the left operand is the first value and the right operand is the second value
   d. For Math, all operands are of data double except for the mod operator which requires both operands be integers.
   e. For Trig, the first value is **d or r** indicating degrees or radians
   f. For Trig, the second value is the angle in degrees or radians

5. **All numbers are of data type double except for the values used with the modulo operator**

6. **Output is to use the default format configuration**, and all output is to the terminal

7. Output is required to match the output provided by the sample solution

8. Use a single loop with a priming read to read the information from the input file. Each loop iteration processes one line from the input file and outputs the results of the calculation.

9. In the loop use nested if-then-else-if statements to process the information in the input file. Or, use nested switch statements. Or, use a combination of switch and if-then-else-if statements.

10. Error messages for the file have 47 characters on the top and bottom lines. The error opening file message has 15 *'s on either side of the title. The empty file message has 13 *'s on either side of title.

11. **For cases of invalid calculation type or invalid operator cases, do not forget to remove any extraneous characters from the input line before processing the next line. Look at input files 1 and 2**

12. **In addition to hint 11, look at input file 3. Your program really just needs to remove any possible extra information after every line is processed – regardless of it being a valid line or one with an invalid calculation or operator.**