SCHOOL OF SCIENCE & TECHNOLOGY

EEET2482 – SOFTWARE ENGINEERING DESIGN

LABORATORY TASK – SIMPLE 2-ARGUMENT CALCULATOR

SPECIFICATIONS

You are required to write a C++ program which performs as a Simple 2-Argument Calculator.
The calculator will need to perform basic calculations as follow.

1. The calculator will continuously take 1-line input from the user as an expression.
2. The calculator will stop when the 1-line input is a single word: '**Exit**'.
3. The expression will be in the form of **[arg1] op [arg2]**. Each part of the expression is separate by single white space.
4. [arg1] and [arg2] will be two integers (could be negative, 0, or positive numbers)
5. op is an operator for the calculation and will be either of the following
    a. '+': performs addition of [arg1] and [arg2]
        i. Given the expression: "2 + 3" the program will return 5
        ii. Given the expression: "2 + -3" the program will return -1
        iii. Given the expression: "-5 + 1" the program will return -4
    b. '-': performs subtraction of [arg2] from [arg1]
        i. Given the expression: "2 - 3" the program will return -1
        ii. Given the expression: "2 - -3" the program will return 5
        iii. Given the expression: "-5 - 1" the program will return -6
    c. '*': performs multiplication of [arg1] and [arg2]
        i. Given the expression: "2 * 3" the program will return 6
        ii. Given the expression: "2 * -3" the program will return -6
        iii. Given the expression: "-5 * 1" the program will return -5
    d. '/': performs division of [arg1] by [arg2]
        i. Given the expression: "2 / 3" the program will return 0
        ii. Given the expression: "2 / -3" the program will return 0
        iii. Given the expression: "-5 / 2" the program will return -2
    e. '%': performs arithmetic remainder of [arg1] divided by [arg2]
        i. Given the expression: "2 % 3" the program will return 2
        ii. Given the expression: "2 % -3" the program will return 2
        iii. Given the expression: "-5 % 2" the program will return -1

From the command line, your program can be executed as followed:

1. *laboratory1_groupTT.exe*
    a. *where TT denotes your group number.*
2. *Once running, the user can input arithmetic expression as follow: arg1 op arg2*
    a. *arg1 is an integer*
    b. *arg2 is another integer*
    c. *op is one of the operator [+, -, *, /, %]*
    d. NOTE: for both arg1 and arg2, the user input arguments can be negative or positive integers.
3. Once user decide to stop the calculator, the user can input the word "Exit".
4. Before the calculator exits, each student ID string must be displayed to the console in the following form.

LABORATORY GROUP TT
sXXXXXXX,sXXXXXXX@rmit.edu.au,FirstName,LastName
sYYYYYYY,sYYYYYYY@rmit.edu.au,FirstName,LastName
sZZZZZZZ,sZZZZZZZ@rmit.edu.au,FirstName,LastName

GENERAL SPECIFICATIONS

ERROR CHECK 1: *Valid Number Input*

If the user input argument(s) cannot be converted into a valid integer(s), your program must display *an appropriate error message* to the console. At this stage, only the conversion to a valid integer is being tested, the integer value may still be found to be out of range.

E.g. 333 and -333 are valid user input arguments

A valid integer will be defined as a number for which the value AFTER the decimal point is zero

Example 1: 33.7, 13000.0004 are NOT valid user input arguments

Example 2: 33. or 33.000 are valid user input arguments

Example 3: +212+21-2 or 5/2 are not valid user input arguments – these are mathematical operations and must be rejected.

You will need to test for other cases, not stated here, for user input arguments which are not valid integers

ERROR CHECK 2: *Number Input Range Check*

If the value of the [arg1] and/or [arg2] argument is NOT in the range of [-32,768 to 32,767], your program must display *an appropriate error message* to the console. NOTE: Zero is a valid user input argument here.

ERROR CHECK 3: *Valid Operator Input*

If the value of the [op] argument is not one of the following [+, -, *, /, %], your program must display *an appropriate error message* to the console.

ERROR CHECK 4: *Division by 0*

If the value of the [arg2] argument is 0 and the value of the [op] argument is '/', your program must display *an appropriate error message* to the console.

ERROR CHECK 5: *Dummy Variable Check*

The correct format is **[arg1] op [arg2]**, so if 1-line input contains dummy variables

For example, 5 + 3 - 4

your program must display *an appropriate error message* to the console.

OTHER SPECIFICATIONS:

1. Your program must be compiled in Microsoft Visual Studio 2017. Programs submitted for assessment which are compiled under different environments (Operating Systems or Development Environments) are not likely to run correctly. If your program does not execute at all, **you will only be eligible for 50% of your laboratory mark**. The teaching staff will NOT be fixing code to make programs compile or for debugging issues during assessment.
2. All code must be written in a single CPP file and must be placed in your report as an appendix.
3. Your group leader, as stated by Canvas, is responsible for submitting the group's work prior to the deadline. **Late submissions will incur a penalty of 10% per day**. **Submissions which are three days past the deadline will not be accepted and a grade of zero will be given**. An executable file of your CPP program, i.e. *laboratory1_groupTT.exe*, and a word document of your report, i.e. *laboratory1_groupTT.docx*, will need to be submitted to Canvas for assessment, where *TT* denotes your group number. Your report will be checked through Turnitin to ensure academic integrity is maintained.
4. You will be required to work in groups of three students.

5. Follow the structure on the following page to write your report.
6. No libraries, except for iostream, can be used – penalties will apply if other external libraries are used.

REPORT STRUCTURE

1. Introduction:
   - Brief literature review on the general laboratory topic – what are the main programming topics and concepts being used in this laboratory?
   - Discuss what you will be doing in this laboratory and relate this to your literature review.
   - Discuss, in words, the structure of the rest of your report.
2. Flowchart
   - Fairly comprehensive diagram which depicts the algorithm you created to write your program.
   - Construct your flowchart from the viewpoint that another person should be able to follow it and write their own software from it.
   - Limit your flowchart to one page.
3. Discussion & Results
   - Clear and concise discussion on all your activities in performing this laboratory.
   - Discuss any issues you came across and how you solved them.
   - Write all your results, in sequence with your discussion, you achieved.
   - Use subheadings (even sub subheadings) to structure this section – the reader should be able to follow with ease what you are discussing.
   - Do not write your software code here, uses references to your software code appendix
4. Conclusions
   - State the main points of this laboratory and how your laboratory work addressed these main points.
   - Should be no longer than 1 paragraph.
   - NOT a section for repeating your activities or re-writing/paraphrasing the laboratory notes.
   - Written in the past tense.
5. References
   - Any references you used must be placed here.
   - IEEE referencing style must be followed.
   - Place citations in your report where necessary.
6. Appendices
   - Your code must be placed here in a separate appendix.
   - Use `Consolas` font and an appropriate font size. Keep the same color format as Microsoft Visual Studio 2017.
   - Other material which you decide to include must go in a separate appendix