RMIT UNIVERSITY

SCHOOL OF SCIENCE & TECHNOLOGY

EEET2482 – SOFTWARE ENGINEERING DESIGN

LABORATORY 2 – DIFFERENTIATOR PROGRAM

SPECIFICATIONS:

You are required to write a C++ program which differentiates an equation that is passed into a console program as a command line argument. The equation can contain any number of terms, where any of the following terms are valid:

Power: $a * x\text{^}(b)$	Logarithmic: $a * log10(x)$	Logarithmic: $a * log10(b * x)$
Exponential: $a * e\text{^}(b * x)$	Logarithmic: $a * loge(x)$	Logarithmic: $a * loge(b * x)$
Exponential: $a * e\text{^}(x)$	Sinusoidal: $a * \sin(x)$	Sinusoidal: $a * \sin(b * x)$
Linear: $a * x$	Co-sinusoidal: $a * \cos(x)$	Co-sinusoidal: $a * \cos(b * x)$
	Constant: a	

Each term can only be separated by a + or – sign and no whitespaces must exist in the equation. In each of the terms, x is the independent variable, a and b are constants and can be assumed to be integer values. An example of how an equation can be is

$$3 * x\text{^}8 + 9 * \log10(8 * x) - 4 * \sin(2 * x) - \log e(x) + 7 * e\text{^}(5 * x)$$

From the command line, your program can be executed in one of the two following ways:

laboratory2_groupXX.exe	*where XX denotes your group number*
laboratory2_groupXX.exe fx	*where XX denotes your group number*

If *ONE* parameter is entered via the command line, *laboratory2_groupXX.exe fx*, where *fx* is the equation and *XX* is your groups number, your program must determine the derivative of the input equation and display the result to the console.

If *TWO* parameters are entered via the command line, *laboratory2_groupXX.exe fx a*, where *fx* is the equation, *a* is an arbitrary input and *XX* is your groups number, your program must exit with an error code where an appropriate statement is output to the console which states the error.

The only output which can be displayed in the command line is a) the resulting equation b) an error message which clearly and concisely states the error which has occurred. NO debug code must appear in the console window.

When your program is run without any user input arguments, i.e. *laboratory2_groupXX.exe*, each student ID string must be displayed to the console in the following form.

LABORATORY GROUP XX
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: *Number of User Input Arguments*

If the number of user input arguments = 0, your program must display the student ID strings in the given form
If the number of user input arguments = 1, the program should proceed to the next check
If the number of user input arguments >= 2, your program must output an appropriate message to the console

ERROR CHECK 2: *Valid Equation Input*

The input equation into the program must be checked for validity. As stated, there must be no whitespaces anywhere in the equation and the constants a and b can be assumed to be integer values only. Only the terms listed on the previous page will be considered, other terms will be considered as invalid input. An example of a valid input equation is also provided on the previous page. Invalid input equations must be rejected and an appropriate message, stating the error, must be displayed 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 across three different files – main CPP file, header file and a source CPP file. Each of these files must be placed in their own appendix at the end of your report. Failure to do this will result in a reduction of 30% of your assignment mark.
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. *laboratory2_groupXX.exe*, and a word document of your report, i.e. *laboratory2_groupXX.docx*, will need to be submitted to Canvas for assessment, where *XX* 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, string and sstream can be used – penalties will apply if other external libraries are used.

You must design your code using object-oriented principles. The following aspects must be considered and implemented

- The terms in the input equation must be stored in a linked-list
- A hierarchy of classes must be constructed which forms all the valid terms in the equation. You will need to consider this part carefully – which terms are more specialized than others? You should also consider intermediate classes in the hierarchy, such as polynomial and trigonometric as two examples.
- Your code must make use of virtual functions for the behaviours which need redefining in the derived classes
- Stroustrup's rule needs to be considered for deciding whether inheritance or aggregation make sense
- If it makes sense to do so, other object-oriented principles should also be considered.

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 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