SCHOOL OF COMPUTING (SOC)

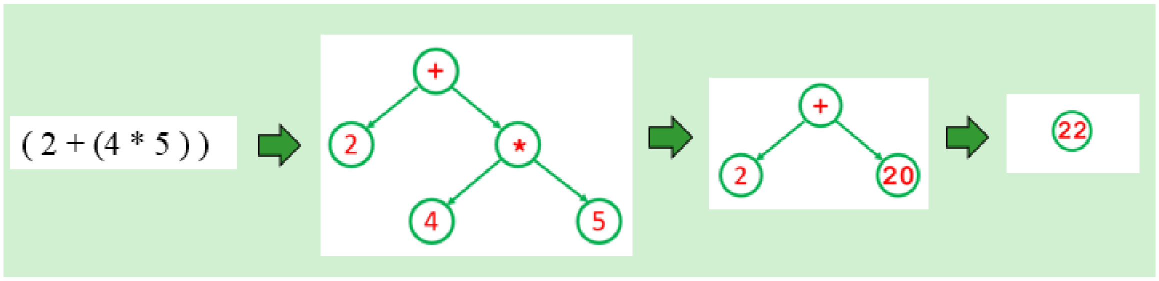
Diploma in Information Technology

ST1507 DATA STRUCTURES AND ALGORITHMS (AI)

2020/2021 SEMESTER 2
ASSIGNMENT TWO (CA2)
~ Expression Evaluator and Sorter (using Parse Trees)~

Objective of Assignment

For this assignment you will have an opportunity to apply all that you have learnt with regards to data structures, algorithms and object oriented programming as to develop an application that can represent, and solve mathematical expressions by making use of parse trees. Parse trees are special binary trees that can be used to represent, and solve mathematical expressions. Below is given an example of a parse tree that has been extracted from an expression, *(2+(4*5))*. The tree can be solved by starting at leaves and then solve the sub-tree expressions first, and progressively work yourself upwards until you reach the root of the tree.



Besides solving individual expressions, your application must also be able to read series of expressions from an input file. These expressions will then need to be evaluated one by one, and sorted by value and length, after which the results will be written back to an output file.

<u>Instructions and Guidelines:</u>

1. This is a group assignment (you will work in pairs, only one group of 3 would be allowed if odd number of total students).

2. This assignment accounts for **40%** of your final grade.

3. The submission date is **Sunday 14 February 11:59 pm**.

4. The development will be carried out in Python using Anaconda.

5. An interim status report need to be submitted in Week 15 (**Sunday 31 Jan, 11:59 pm**)

6. The interviews will be conducted during the DSAA lessons in week 18. You are expected to explain on your code and program logic. Take note that the interview is compulsory. In case you are absent from the interview without valid reasons, you will not get any marks for the assignment.

7. No marks will be given, if the work is copied, or you have allowed others to copy your work.

8. **50% of marks** will be deducted for submission of assignment within **ONE** calendar day after the deadline. **No marks shall** be awarded for assignments submitted **more than one day** after the deadline.

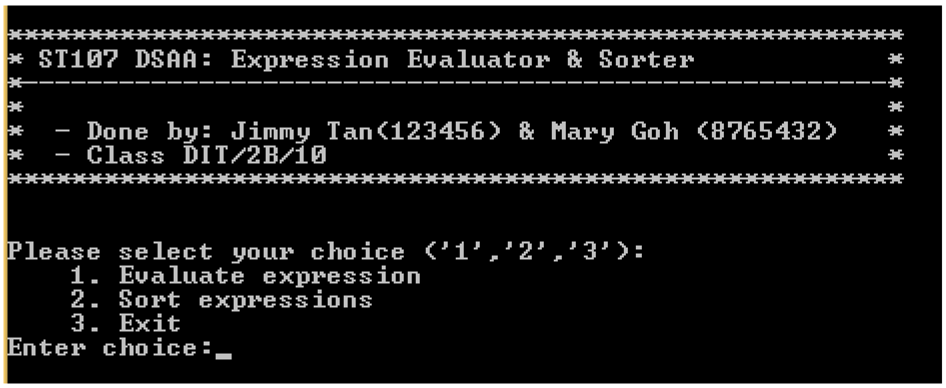
Warning: Plagiarism means passing off as one's own the ideas, works, writings, etc., which belong to another person. In accordance with this definition, you are committing plagiarism if you copy the work of another person and turning it in as your own, even if you would have the permission of that person. Plagiarism is a serious offence and disciplinary action will be taken against you. If you are guilty of plagiarism, you may fail <u>all</u> modules in the semester, or even be liable for <u>expulsion</u>.

Overview of the system

Your job is to implement an *Expression Evaluator and Sorter*. The application should be able to represent, print, and solve fully parenthesized mathematical expressions by means of parse trees. Besides solving individual expressions, your application must also be able to read a series of expressions from an input file. These expressions will then be evaluated one by one, sorted by their value and length. Subsequently the sorted results will be written back to an output file.

Selection menu

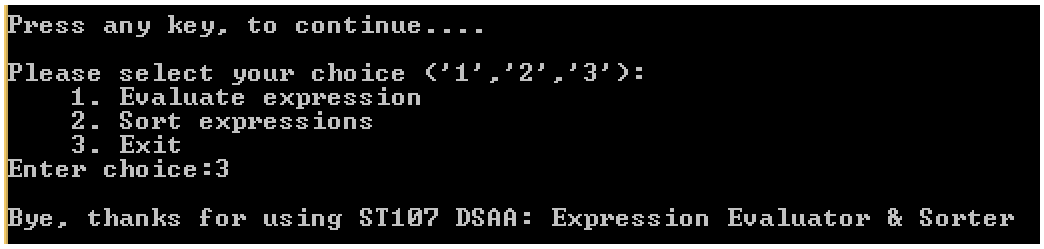
When the application starts, the user will be presented with a menu as is shown below, allowing the user to choose from 3 options ('1','2','3').



```
**********************************************************
* ST107 DSAA: Expression Evaluator & Sorter              *
*--------------------------------------------------------*
*                                                        *
*  - Done by: Jimmy Tan(123456) & Mary Goh (8765432)     *
*  - Class DIT/2B/10                                     *
**********************************************************


Please select your choice ('1','2','3'):
    1. Evaluate expression
    2. Sort expressions
    3. Exit
Enter choice:_
```

Take note you must follow the above format, please ensure you display names and IDs of all group members, as well as the correct class.

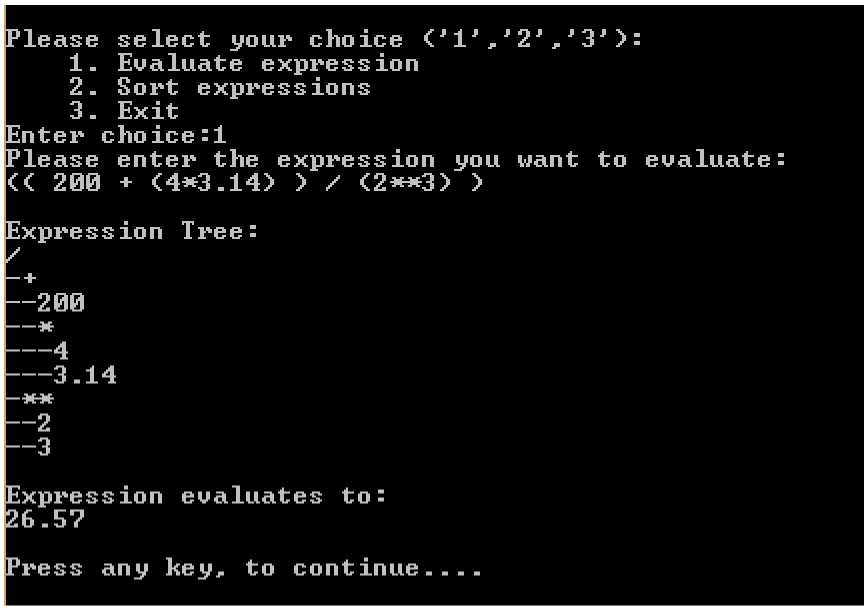
The user will be able to repeatedly select options from the menu, until he/she selects option 3 after which the application will terminate.



```
Press any key, to continue....

Please select your choice ('1','2','3'):
    1. Evaluate expression
    2. Sort expressions
    3. Exit
Enter choice:3

Bye, thanks for using ST107 DSAA: Expression Evaluator & Sorter
```

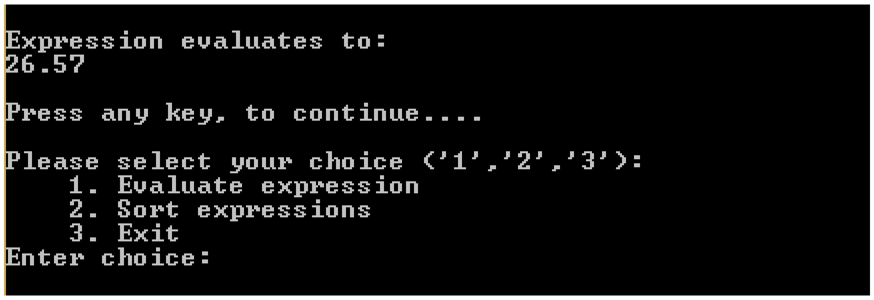
Evaluating Expressions

When the user selects option ‘1’, he/she will be prompted to enter a fully parenthesized expression. The application will then calculate, and print the parse tree followed by displaying the value that the expression evaluates to. The printing of parse tree is to done in preorder traversal, with the number or dashes (e.g. ‘-‘) indicating the depth of a node.



```
Please select your choice ('1','2','3'):
    1. Evaluate expression
    2. Sort expressions
    3. Exit
Enter choice:1
Please enter the expression you want to evaluate:
(( 200 + (4*3.14) ) / (2**3) )

Expression Tree:
/
-+
--200
--*
---4
---3.14
-**
--2
--3

Expression evaluates to:
26.57

Press any key, to continue....
```

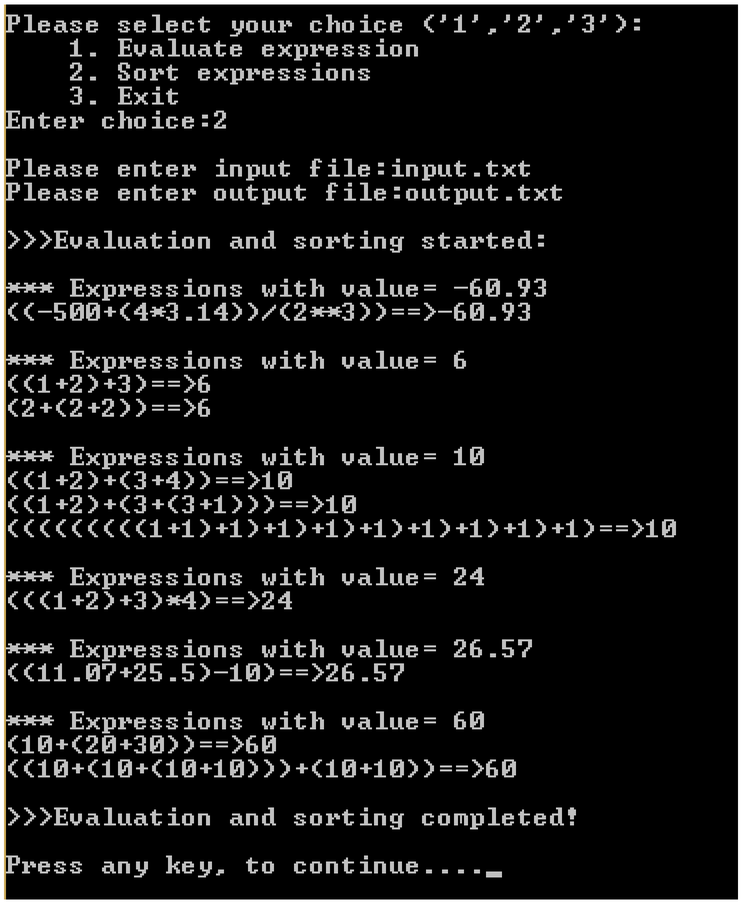
After the uses presses a key to continue, the same menu will be displayed again, and the user may either evaluate another expression, or proceed using another option.



```
Expression evaluates to:
26.57

Press any key, to continue....

Please select your choice ('1','2','3'):
    1. Evaluate expression
    2. Sort expressions
    3. Exit
Enter choice:
```

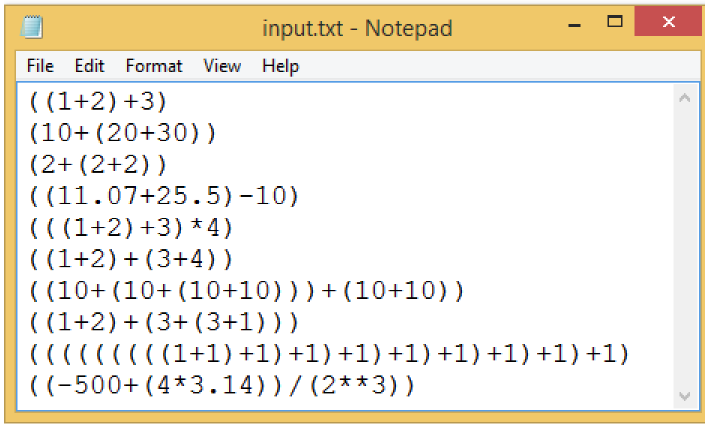
Sorting Expressions

When the user selects option ‘2’, he/she will be prompted to enter an input file and output file. The application reads the expressions, from the input file, evaluates each expression (using a parse tree), and then sort the expression first by value (in ascending order) and then by length (in ascending order). The output will be both printed on the screen as well written to the output file.

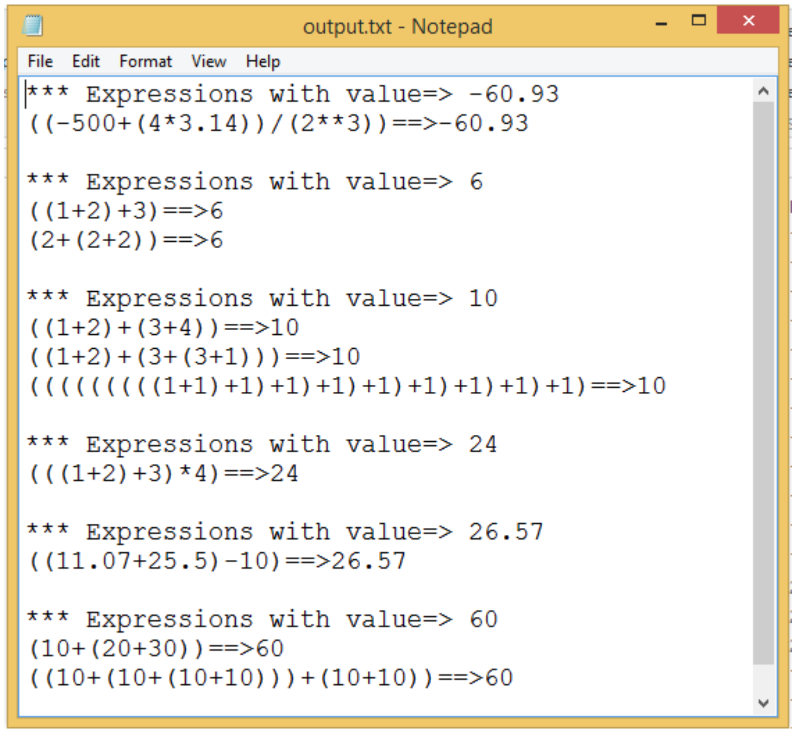


```
Please select your choice ('1','2','3'):
    1. Evaluate expression
    2. Sort expressions
    3. Exit
Enter choice:2

Please enter input file:input.txt
Please enter output file:output.txt

>>>Evaluation and sorting started:

*** Expressions with value= -60.93
((-500+(4*3.14))/(2**3))==>-60.93

*** Expressions with value= 6
((1+2)+3)==>6
(2+(2+2))==>6

*** Expressions with value= 10
((1+2)+(3+4))==>10
((1+2)+(3+(3+1)))==>10
(((((((((1+1)+1)+1)+1)+1)+1)+1)+1)+1)==>10

*** Expressions with value= 24
(((1+2)+3)*4)==>24

*** Expressions with value= 26.57
((11.07+25.5)-10)==>26.57

*** Expressions with value= 60
(10+(20+30))==>60
((10+(10+(10+10)))+(10+10))==>60

>>>Evaluation and sorting completed!

Press any key, to continue....
```

Next are both a screen shot of an example input file, and the associated output file:



```
((1+2)+3)
(10+(20+30))
(2+(2+2))
((11.07+25.5)-10)
(((1+2)+3)*4)
((1+2)+(3+4))
((10+(10+(10+10)))+(10+10))
((1+2)+(3+(3+1)))
(((((((((1+1)+1)+1)+1)+1)+1)+1)+1)+1)
((-500+(4*3.14))/(2**3))
```



```
*** Expressions with value=> -60.93
((-500+(4*3.14))/(2**3))==>-60.93

*** Expressions with value=> 6
((1+2)+3)==>6
(2+(2+2))==>6

*** Expressions with value=> 10
((1+2)+(3+4))==>10
((1+2)+(3+(3+1)))==>10
(((((((((1+1)+1)+1)+1)+1)+1)+1)+1)+1)==>10

*** Expressions with value=> 24
(((1+2)+3)*4)==>24

*** Expressions with value=> 26.57
((11.07+25.5)-10)==>26.57

*** Expressions with value=> 60
(10+(20+30))==>60
((10+(10+(10+10)))+(10+10))==>60
```

Take note, you are required to follow the output format.

Basic requirements:

- Your application will only need to work for fully parenthesized expressions. Do take note that the placement of the parentheses will dictate the order that the operators are being executed. So for instance the following 3 expression will be evaluated to different values as is dictated by the placement of the parentheses.

```
((1+2)/(3*5))        ==>0.2
(1+((2/3)*5))        ==>4.33
(((1+2)/3)*5)        ==>5.0
```

- You need to support at least the following operators:

Operator	**Description**
+	*addition*
-	*subtraction*
*	*multiplication*
/	*division*
**	*exponent*

- You will need to support both integer as well as float operands. Operands can be both positive or negative.

- You are required to design, and write the Python application using an object oriented approach (OOP). You should leverage on your knowledge of encapsulation, polymorphism, inheritance etc.

- You may make use of Python's already build in data structures, such as list, tuple, dictionary etc., however you should refrain of using the classes from the collection library. Instead you are required to write you own classes to support the various data structures that you may need (for instance *Tree*, *SortedList* or *Stack* etc.). Of course you may refer back to the lecture slides and lab tasks, and expand further on those classes that we had previously developed and discussed in tutorial and lab sessions.

- The classes that you develop must be placed in separate python files.

- Pay attention to user input validation. Your application should not crash if a user types in the wrong input. Instead when a user enters wrong input, notify the user, and allow him/her to enter again.

Additional features:

After you have implemented the basic requirements you may consider implementing additional features such as (but not limited to) the following:

- Ability to support additional operators other than the required ones.
- Ability for user to choose between sorting in ascending or descending order.
- Ability to provide other sorting criteria (other than value & length)
- Ability to choose alternative ways of printing the parse tree beside required pre-order (for instance you may consider a post-order or in-order print).
- Any other features that may enhance the system.

(*) You may score up to a maximum of 10/100 marks for additional features.

<u>Interim Status report</u>

You are required to submit an interim status report in week 15. You need to submit the interim report in BB by **Sunday 31 Jan, 11:59 pm**. Take note, late submission will not be entertained.

Submit the interim report, as a pdf file with a <u>maximum</u> **of 5** pages. This excludes cover page and the appendix with the source listing, and references. The report should contain:

a) Cover page with names, ids and class of team
b) Discussion on the various classes that you plan, or have already developed, for your application. You may discuss on issues such as, how you plan to apply Object Oriented Technology (OOP) to develop the classes, and how do the various classes relate together.
c) Describe the challenges the team is facing, this could be technical challenges, as well as challenges that comes with working as a team. Mention the challenges that you already have overcome, and list those challenges that still lay ahead.
d) Include a clear description of the roles and contributions of each member in the team. Clearly state what programming tasks each member has/will been responsible for. Mention what programming work has already been carried out, and what is still to be done.
e) Include in the appendix any <u>source code listings</u> of the work in progress that you may already have developed in this point of time. Clearly indicate who wrote what code. You may also include in the appendix, the various references from literature or internet that you may have /plan to consult.

Final Deliverables

Your final deliverables must include:

(a) Final Report

A report (as pdf file) with a maximum **of 5** pages. This excludes cover page and the appendix with source listing, and references. The report should contain:

a) Cover page with names, ids and class of team
b) Description, and user guidelines, as on how to operate your application (please include screen shots of your application in action).
c) Discussion on the various classes that you have developed for your application. You may discuss on issues such as, how did you applied Object Oriented Technology (OOP), challenges faced, solutions found to overcome the challenges. Include a summary of the key take always and learning achievements you have obtained from this project.
d) A clear description of the roles and contributions of each member in the team. Clearly state what each member has been responsible for, and what programming work has been carried out by each member.
e) All your python source code listings must be included as an appendix at the end of your report. You must clearly indicate who wrote what code. You may also include in the appendix, those references from literature or internet that you may have consulted.

(b) Source Code

- You must submit all the python files (.py files) that makes up your application. Make sure the code is complete, and can run directly from the Anaconda Prompt

Submission deadline

Final Submission is on **Sunday 14 February 11:59 pm**.

-Submit all the deliverables (Source Code, Report) in the designated Blackboard drop box before the deadline.

- You must submit it as one Zipped folder (RAR will not be accepted, only zip) whereby you label your submission as:

CA2_Final_ StudentID1_StudentID1Class.zip

For example: *CA2_12345_8765_DIT2B02.zip*

Assessment Criteria

The assignment will be assessed based on the following criteria:

Assessment criteria	**Marks awarded**
File IO & GUI: - Displays names, id's and class as prescribed. - GUI with menu that allows user to repeat actions until user decides to exit. - Supports reading expressions, and writing expressions to file.	Max 15
Expression Parsing and Evaluation: - Expressions correctly parsed and evaluated through a parse tree. - Parse tree is correctly printed.	Max 10
Expression Sorting: - Expression are correctly sorted by values and length - Sorted expression are displayed, and written in the correct output format	Max 10
Programming techniques and readability of code: - Proper usage of object oriented programming techniques as to implement your data structures and application. - Functions and programming constructs. - Comments. - Free of crashes.	Max 25
Interim Status Report: - You must follow the prescribed format	Max 10
Final Report: - You must follow the prescribed format -	Max 10
Interview: - Ability to demonstrate and explain the application. - Q&A	Max 10
Extra features	Max 10
Grand Total	**100**

-- End --