**SCHOOL OF COMPUTING (SOC)**

Diploma in Applied AI and Analytics

ST1507 DATA STRUCTURES AND ALGORITHMS (AI)

2021/22 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 the leaves and then solve the sub-tree expressions first, and progressively work yourself upwards until you reach the root of the tree.
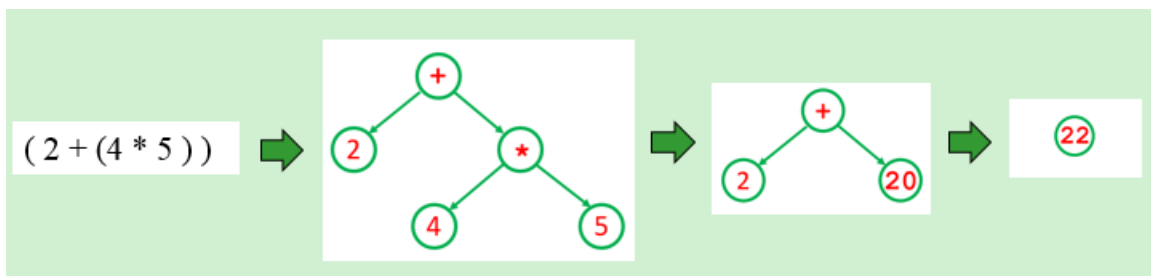
Besides solving individual expressions, your application must also be able to read series of expressions from an input file. These expressions will then need to be evaluated one by one, and sorted by value, length and number of brackets, after which the results will be written back to an output file.

Instructions and Guidelines:

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

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

3. The assignment comprises a group component (70%) and an individual component (30%).

4. The submission date is <mark>**Wednesday 9 February 1:00 pm**</mark>.

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

6. The demonstrations/interviews will be conducted during the DSAA lessons in week 17/18. 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 all modules in the semester, or even be liable for expulsion.

Overview of the basic features 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').

```
*********************************************************
* ST1507 DSAA: Expression Evaluator & Sorter           *
*-----------------------------------------------------*
*                                                       *
*   - Done by: Jimmy Tan(123456) & Mary Goh (8765432)   *
*   - Class DAAA/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 the 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 ST1507 DSAA: Expression Evaluator & Sorter
```

Option 1: 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 be done in **in-order** traversal, with the number of dots (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:
(2+(4*5))

Expression Tree:
..5
.*
..4
+
.2
Expression evaluates to:
20

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:
20

Press any key, to continue....

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

Option 2: 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 <u>ascending</u> order) and then by length (in <u>ascending</u> order). If expressions have same values and same length they will be sorted by total number of brackets (in <u>descending</u> order).
- If there are any spaces in an expression then they will be removed.
- When calculating the length of an expression you must exclude spaces.
- The output will be both printed on the screen as well as written to an 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)==>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...._
```
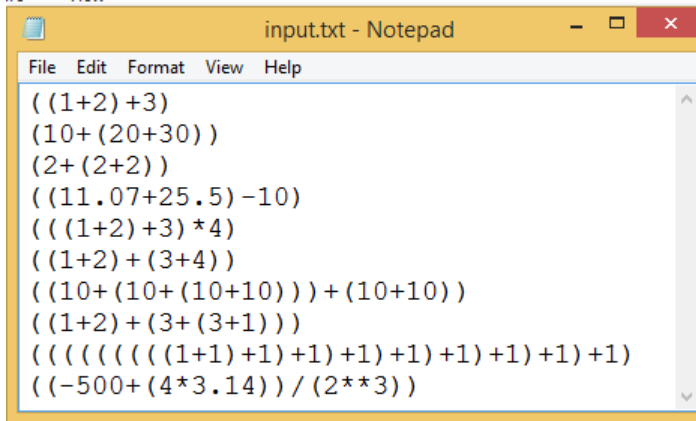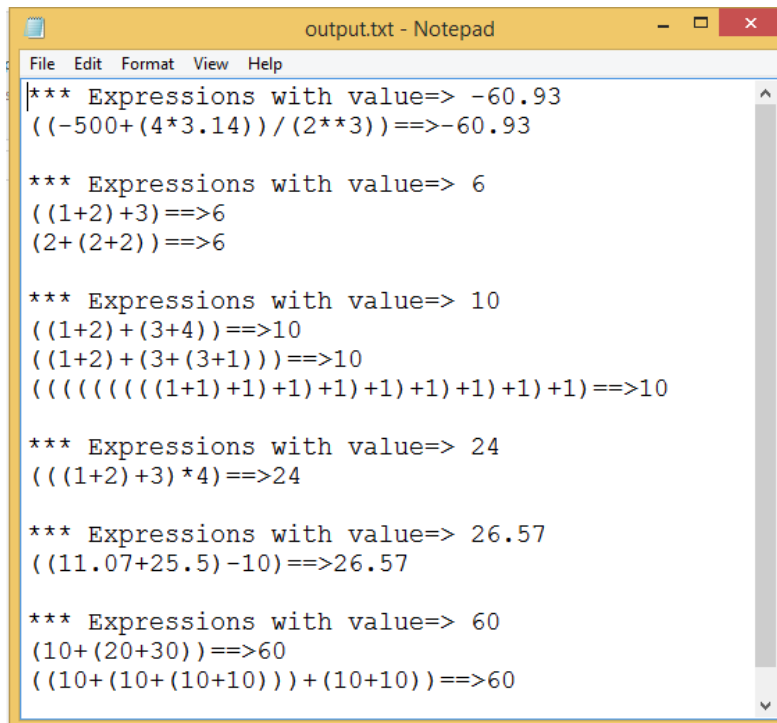
Next are both a screen shot of an example input file, and the associated output file:

```
                    input.txt - Notepad          –  □  ×
File  Edit  Format  View  Help
((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)
((-500+(4*3.14))/(2**3))
```

```
                    output.txt - Notepad          –  □  ×
File  Edit  Format  View  Help
*** 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)==>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 above output format.

Option 3: Exit

The user can repeatedly select Options 1and 2. Option 3 is to exit the program. Take note that after the group adds additional options for extra features, Option 3, to exit will be shifted as the last option (e.g., it will remain the last option).

Requirements for Group Component (70%):

- Your application will only need to work for <u>fully</u> 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
```

- Your application needs to support the following 5 operators:

$$+, -, *, /, **$$

- The application will need to support <u>two</u> groups of different implementations for the operators:

Operator Implementation Group One:

Operator	Description	Examples
+	*A regular addition.*	(1+2) => 3
-	*A regular subtraction.*	(3-1) => 2
*	*A regular multiplication.*	(2*3) => 6 (2*3.14) => 6.28 (4*3.14) => 12.56
/	*A regular division.*	(1/4) => 0.25 (3/4) => 0.75 (3.14/2) => 1.57
**	*A regular exponent.*	(2**3) => 8 (3**2) => 9

Operator Implementation Group Two:

| Operator	Description	Examples
+	*Takes maximum of two.*	`(100+200) => 200`
-	*Takes minimum of two.*	`(50-200) => 50`

*	*Multiplies, and then rounds to closest integer*	`(2*3) => 6`
		`(2*3.14) => 6`
		`(4*3.14) => 13`

/	*Divides, and then rounds to closest integer.*	`(1/4) => 0`
		`(3/4) => 1`
		`(3.14/2) => 2`

**	*Returns remainder after division (works as % e.g., as the modulus operator).*	`(2**3) => 2`
		`(3**2) => 1`
		`(21**7) => 0`
		`(25**7) => 4`

- The user will be able to alternate between the two Operator Implementation Groups by means of a configuration file (named *config.txt*).
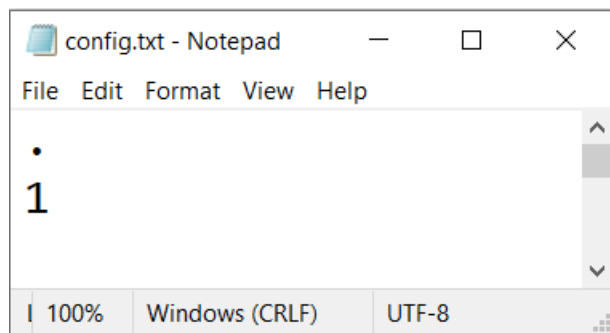
- At start time the application will read the setting from this configuration file (*config.txt*), The first line contains the separator (used for indicated the depth while printing the tree), and the second line contains the operator implementation group, either a '1' (Operator Group One) or '2' (Operator Group Two). The default setting for separator is '.'. The default setting for Operator Implementation Group is '1' (Operator Implementation Group One).

- Take note the separator is meant only to be one character, and it may also be a space character (e.g. ' ').

Example 1:
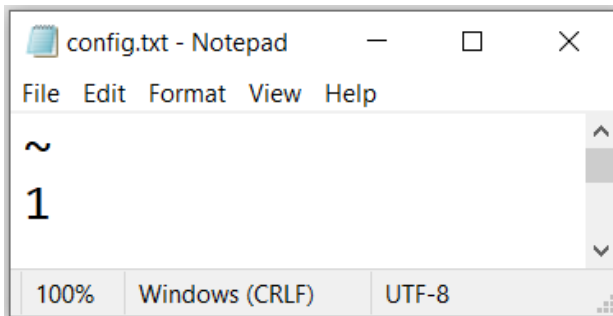
```
config.txt - Notepad        —    □    ×

File  Edit  Format  View  Help

~

1

100%    Windows (CRLF)      UTF-8
```

```
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:
~~3
~**
~~2
/
~~~3.14
~~*
~~~4
~+
~~200
Expression evaluates to:
26.57

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

Example 2:
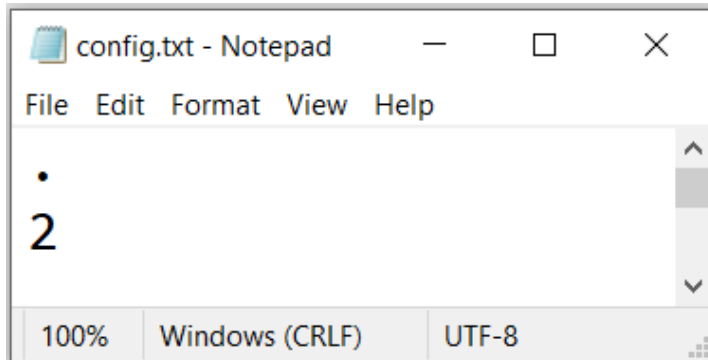
config.txt - Notepad
File Edit Format View Help

.
2

100%   Windows (CRLF)   UTF-8

```
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:
..3
.**
..2
/
...3.14
..*
...4
.+
..200
Expression evaluates to:
100

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

- You will need to support both <u>integer</u> as well as <u>float</u> operands. Operands can be either <u>positive</u> or <u>negative</u>.

- 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 and set. 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 to the lecture slides and lab tasks and expand further on those classes that we had previously developed in the tutorial and lab sessions.

- The OOP 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.

- The group will be requested to demonstrate the basic features of the application during the demonstration.

- Take note the group's demonstration should not exceed 15 minutes (including 5 minutes for Q&A).

Requirements for Individual Component (30%):

Each individual team member is required to implement <u>two additional features</u> to be added to the application. These two additional features will need to be presented during the final presentation.

- The two features added must be integrated in the form of additional menu items in the application.
- The features will be graded on technical sophistication, uniqueness, innovativeness, and usability.
- Take note features within the same group must be different. So please check with you group member(s) first before embarking on implementing the extra features.
- Each group member must present his/her features through a short PowerPoint presentation followed by a demonstration of the features (total presentation time not to exceed 10 minutes)
- Your PowerPoint slides must briefly describe what features you have implemented. You must include screen shots demonstrating the features in action. Please explain how the features work, and why they are useful. You are to include your Name, Class and Group Number in the first slide.
- You must submit the PowerPoint Slides (converted to pdf) together with a compulsory Peer Feedback (template will be provide on BlackBoard).

Final Deliverables

Your group's final deliverables must include:

(a) Group Report

A report (as pdf file) with a <u>maximum</u> **of 10** pages. This excludes cover page and the appendix with source listing and references. The report should contain:

a) Cover page with group number (your instructor will assign group numbers) names, ids, and class.
b) Description, and user guidelines, on how to operate your application (please include screen shots of your application in action).
c) Describe how you have made use of the Object-Oriented Programming (OOP) approach. You may elaborate of the classes that you have developed, and discus on issues such as encapsulation, function/operator overloading, polymorphism, inheritance etc. Include a class diagram that displays the relation between the various classes you have developed.
d) Discussion on the data structures and algorithms you have developed for your application. You may discuss on issues such as, the performance of the algorithms in terms of Big (O). Explain why you did develop certain data structures and explain why you deem these data structures suitable for the task(s) at hand. Include a table summarizing all the data structures that you have been using (those that you have developed and those already build in Python).
e) Include a summary of the challenges that the group has faced while developing the application (that should include both technical, as well as group-work challenges). Provide a summary of the key take aways and learning achievements that you have obtained from this project.
f) Include 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.
g) **All** your python **source code listings** must be included as an appendix at the end of your report. You must clearly indicate in the source code listings <u>who wrote what code</u>. You may include in the appendix, those references from literature or internet that you may have consulted.

(b) Source Code

- You must submit <u>all</u> (*) the python files (.py files) that makes up your application. Ensure code is complete, and that it can run directly from the Anaconda Prompt.

    (*) Take note, that includes the code for <u>all</u> the extra features as well.

Submission instructions

Group Submission:

- Group Leader to submit all the group deliverables (Source Code and Group Report) in the designated Blackboard Drop Box.

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

   CA2_GroupNumberClass.zip

   For example: *CA2_GR_10_DAAA_2B08.zip*

- Please ensure that you submit it by the stipulated deadline.

Individual Submission:

- Each individual Group Member is to submit his/her individual deliverables.

- Individual submission to include the PowerPoint slides (converted to pdf) and the Peer Feedback. Please ensure that you submit it in the designated Blackboard Drop Box for individual submissions.

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

   CA2_Final_ GroupNumberNameClass.zip

   For example: *CA2_GR_10_JIMMY_TAN_DAAA_2B08.zip*

- Please ensure to submit it by the stipulated deadline.

Assessment Criteria

The group component of the assignment will be assessed based on the following criteria:

Assessment criteria (Group 70 %)	Marks awarded
GROUP COMPONENT (70 %)	
File IO & GUI: - GUI with a menu that operates as is prescribed and has appropriate user input validation and error handling. - Supports reading and writing to files with appropriate user input validation and error handling.	Max 10
Expression Parsing and Evaluation: - Expressions are correctly parsed and evaluated through a parse tree. - Parse trees are correctly printed.	Max 10
Expression Sorting: - Expressions are correctly sorted by values, length, and number of brackets. - Sorted expressions are correctly displayed and written in the required output format.	Max 10
Programming techniques and efficiency: - Appropriate usage of classes and OOP technology. - Appropriate usage of data structures and algorithms.	Max 10
Robustness, and readability of code: - Code is properly commented and neatly structured. - Application is free of crashes.	Max 10
Group Report: - Report follows the prescribed format. - Report is well written and comprehensive.	Max 10
Group's demonstration: - Group effectively demonstrates the basic features. - Group's ability to answer questions raised in Q&A.	Max 10
Group Total	**70**

The individual component of the assignment will be assessed based on the following criteria:

Assessment criteria (Group 70 %)	Marks awarded
INDIVIDUAL COMPONENT (30 %)	

Extra Feature One
   -   Technical sophistication.
   -   Uniqueness/innovativeness/usability.
	Max 10

Extra Feature Two
   -   Technical sophistication.
   -   Uniqueness/innovativeness/usability.
	Max 10

Presentation & Demonstration:
   -   PowerPoint slides.
   -   Demonstration of features and Q&A.
	Max 10

Individual Total	**30**

(*) Take note in case of group members with poor contribution to the group effort a multiplier may be applied (peer feedback may be taken in consideration).

~ End ~