**ST1507 DSAA ASSIGNMENT TWO (CA2)**

**EXPRESSION EVALUATOR & SORTER**

GROUP NUMBER: 5

NAMES: IVAN TAY YUEN HENG & CHAN JUN YI

STUDENT IDS: 2335133 & 2309347

CLASS: DAAA/FT/2A/21

## User Guidelines

Press Enter to continue. A list of options will then be displayed, allowing you to select a choice between numbers 1 and 7 only. No other choices will be accepted.

### 1) Evaluate expression

### 2) Sort expressions

### 7) Exit

## Object-Oriented Programming (OOP) approach

### Encapsulation

A screen shot of a computer program

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.

I implemented encapsulation by using attributes like \_\_file\_content(), \_\_file\_name, and \_\_output\_file, making them inaccessible directly from outside the class. Controlled access is provided through methods such as get\_content(), get\_filename(), and get\_output\_file\_name(). This ensures the internal state remains secure, allowing external code to retrieve data without the risk of unintended modifications.

### Operator Overloading

A computer screen with text on it

AI-generated content may be incorrect.

I overloaded the \_\_lt\_\_ method in the Node class to sort expressions based on three criteria. First, expressions are compared by their evaluated result (data[1]). If results are the same, shorter expressions (len(data[0])) take priority. If both are equal, the expression with fewer parentheses is prioritized. This ensures sorting aligns with the assignment requirements.

### Polymorphism

A black screen with text

AI-generated content may be incorrect. A black and white text

AI-generated content may be incorrect.

I implemented polymorphism by defining a base class, Tree, with an abstract method that is overridden in its subclasses, BinaryTree and ExpressionTree. Each subclass provides its own unique implementation, ensuring that the method behaves differently based on the specific subclass.

### Inheritance

A computer screen shot of a black background

AI-generated content may be incorrect. A computer code with text

AI-generated content may be incorrect.

I implemented inheritance with the BinaryTree and ExpressionTree classes, both of which inherit from Tree. The Tree class defines a common attribute, myStack, and super().\_\_init\_\_() is used to initialize properties from the Tree class.

### Class Diagram

## Data structures and algorithms

|  |  |
| --- | --- |
| Data structure | Big (O) |
| Binary Tree | Insert(Left or Right): O(log n), O(n) (worst case)  Search: O(log n) average, O(n) (worst case)  Space: O(n) |
| Sorted List | Insert: O(n)  Search: O(n)  Sorting Complexity: O(n) |
| Stack | Push/Pop/Peek/Get Size: O(1)  Space complexity: O(n) |

### Binary Tree

A screen shot of a computer program

AI-generated content may be incorrect.

A Binary Tree enables O(log n) insertions and searches when balanced but degrades to O(n) in the worst case if unbalanced. It structures mathematical expressions hierarchically, ensuring correct operation precedence and correct evaluation

### Stack

A screen shot of a computer program

AI-generated content may be incorrect.

The Stack provides O(1) time complexity for push, pop, and checking if empty, making it efficient for last-in, first-out (LIFO) operations. It helps handle nested expressions especially in creating binary tree in (option 1).

### Sorted List

A screen shot of a computer program

AI-generated content may be incorrect.

Sorted Lists maintain elements in sorted order (O(n)) as they are inserted, using the Node class to compare values while storing n nodes (O(n)), ensuring expressions are correctly ordered for option 2 of your assignment.

## Summary

A close-up of a white background

AI-generated content may be incorrect.

### Challenges

### Key takeaways and learnings

## Roles and contributions

### IVAN TAY YUEN HENG (2335133)

### For Option 1, implement the BuildParseTree class, allowing users to construct a binary tree and validate mathematical expressions using the ExpressionTree class, with both classes inheriting from a common parent class, Tree. Develop a GUI interface to ensure user are able to input and visualise expressions.

### For Option 2, create the ReadFile and OutputFile classes to handle user input and output file operations required for sorting expressions. Implement the SortExpressions class, which evaluates mathematical expressions and stores them in a SortedList. Implement Node class to sort them based on three criteria: descending order of expression value, ascending order of expression length, and ascending order of bracket count

### Contribute to the group report, covering the OOP approach and data structure & alogrithms with explanations and screenshots and the proper formatting of the final report layout.

### The project that Ivan created: ( binaryTree.py, buildParseTree.py, expressionTree.py, fileHandling.py, fileOutput.py, sortedList.py, sortedNode.py, sortExpression.py, stack.py, tokeniser.py, and tree.py )

### CHAN JUN YI (2309347)

Roles:

## Appendix