Faculty of Engineering **جامعة الإسكندرية**

Specialized Scientific Programs **كلية الهندسة**

Computer and Communications Dept **البرامج العلمية المتخصصة**

Subject: Data Structures CC273

Simple Interpreter Report

User Manual Guide

Team Members:

Mohannad Bashar – 6656

Mohammed Zayton – 6670

Yousef Khater – 5825

Youssef Victor – 6668

Project Description: *In this project, we have implemented an interpreter that executes instructions and then prints out the results of the execution of these instructions. The interpreter accepts the input either through a text file or writing directly in the GUI text editor, then the input is parsed and the value of each numeric variable is printed. After going through the parser, the variables are sorted in two ways: by value and by variable name.*

Part I: Parsing

At the launch and execution of the code the user is prompted to enter the source text file to be interpreted. The input is read line-by-line. Each line go through a parsing process. In this step, the interpreter determines the type of statement (assignment statement, if statement, or goto statement) and determines whether it contains a label declaration or not. After that, this statement is executed and the variable value is set in a HashMap. During the execution of a statement, any expected expressions are sent to the expression evaluator. The expression evaluator converts from infix expression to postfix, then evaluates the postfix expression. The conversion from infix to postfix is done using the Shunting-yard algorithm. For more information about the algorithm, see the following link.

`https:// wikipedia.org/wiki/Shunting-yard_algorithm`

The expression evaluator supports both unary and binary operators.

Part II: Sorting

After the input is executed, it's time to do the sorting.

There are two variants when it comes to sorting the expression, the first is to sort the variable based on its alphabetical precedence, and the second is to sort the value based on its numerical precedence, each with their own sorting algorithm.

When it comes to the sorting by name, we use binary search tree (BST) in order traversal. The way that in-order traversal works is as follows. Each node in our BST contains a key (variable name) and a value (variable value) and the way it is filtered is by using in-order which filters the tree using recursion based on the following precedence, Left -> Root -> Right. Insertion of a new item is also done with the BST by inserting a key and a value to a new node, or if the key already exists the value is then updated and after a new node is inserted we rerun the code and sort the tree by value

When it comes to sorting by value we use the heap sorting algorithm.

In heap same BST node is used, minimum value which is the root is extracted then heapify is applied and the extracted minimum assigned to the heap last element, looping for all elements. Finally, we got a sorted heap.