COMP 1123
Lab 9

In this lab, your first task is converting valid expressions from infix to postfix notation using linked stacks. After that, you will evaluate the postfix expression.

1.Conversation

Suppose we want to convert the infix expression

```
a + b * c  + ( d * e + f ) * g
```

Into its postfix correspondent.

- We start with an initially empty stack.
- When an operand is read, it is placed onto the output.
- Operators and left parenthesis are pushed to a stack.
- If we see a right parenthesis, then we pop the stack, writing symbols until we encounter a left parenthesis, which is popped but not output.
- If we see any other symbol i.e. +, *, (, then we pop entries from the stack until we find an entry of lower priority.
- One exception is that we never remove a '(' from the stack except when processing a ')'.
- When the popping is done, we push the operand onto the stack.
- Finally, if we read the end of input, we pop the stack until it is empty, writing symbols onto the output.

Final output :

```
a b c * + d e * f + g * +
```

2.Postfix Expression Evaluation

The postfix expression evaluation algorithm takes as an input to the above algorithm output. To start the evaluation part, you must first develop an algorithm that is capable of converting a valid expression infix to postfix.