**TABLE OF CONTENTS**

1. Introduction ........................................... 1

2. Program Design and Logic ............................... 2

3. Input and Output File Usage ............................ 3

4. Testing and Results .................................... 4

5. Conclusion ............................................. 5

6. References ............................................. 6

**Introduction**

This assignment focuses on implementing a program that reads infix mathematical expressions from a file, converts them to postfix notation using stack-based logic, evaluates the result, and writes the output to a file. The goal is to demonstrate understanding of data structures, particularly stacks and queues, and file handling in Python.

**Program Design and Logic**

**Language Used: Python 3**

**Key Concepts Applied:**

• Stack for infix-to-postfix conversion

• Stack for postfix evaluation

• File I/O for reading and writing data

**Structure:**

• : Reads expressions from

• : Converts infix to postfix using operator precedence

• : Evaluates postfix expressions

• : Writes results to

**Input and Output File Usage**

**input.txt**

• Contains one infix expression per line

• Example:

3 + 4 \* (2 - 1)

(5 + 6) \* 2

**output.txt**

• Contains evaluated results, one per line

• Formatted to 2 decimal places

• Example:

7.00

22.00

**Testing and Results**

**Test Cases:**

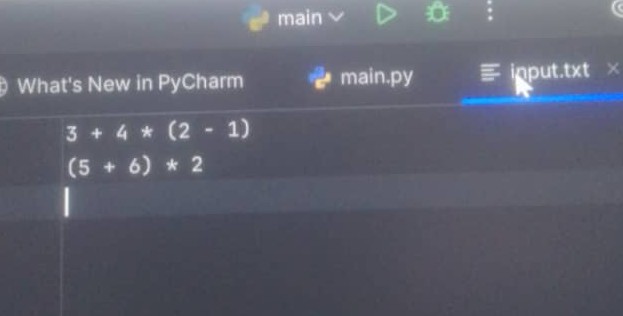
|  |  |  |
| --- | --- | --- |
| Infix Expression | Postfix Expression | Result |
| 3 + 4 \* (2 - 1) | 3 4 2 1 - \* + | 7.00 |
| (5 + 6) \* 2 | 5 6 + 2 \* | 22.00 |
| 10 / (3 + 2) | 10 3 2 + / | 2.00 |
| 8 - 3 + 2 | 8 3 - 2 + | 7.00 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

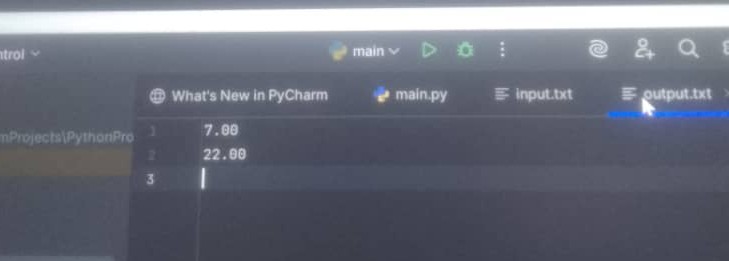
**Screenshots :**

* Screenshot of input.txt
* Screenshot of output.txt
* Screenshot of PyCharm running the code

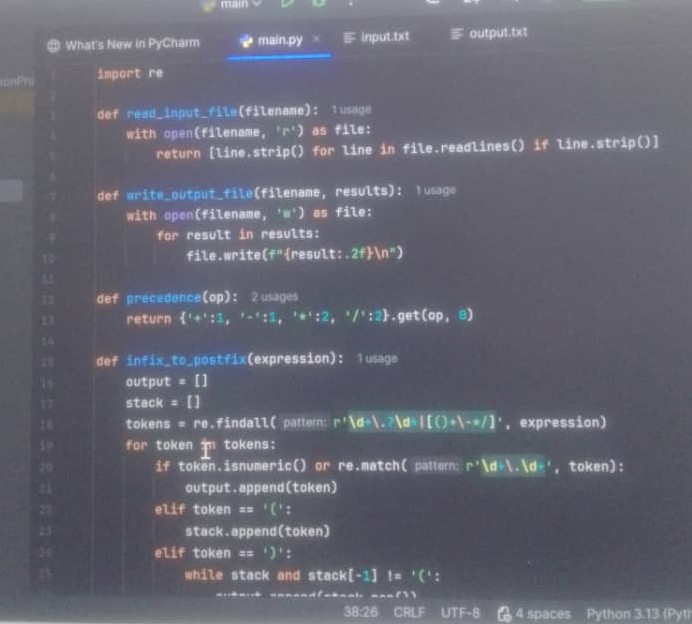
**Input.txt screenshot**

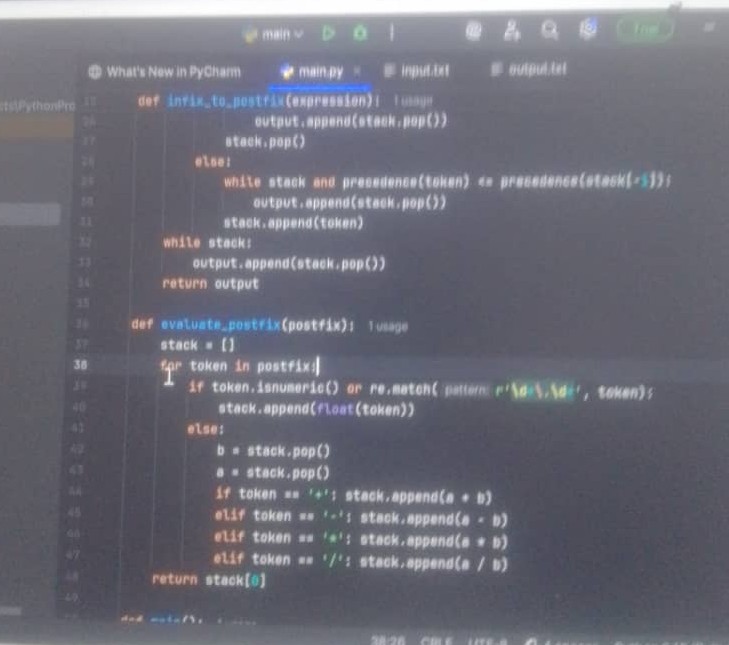


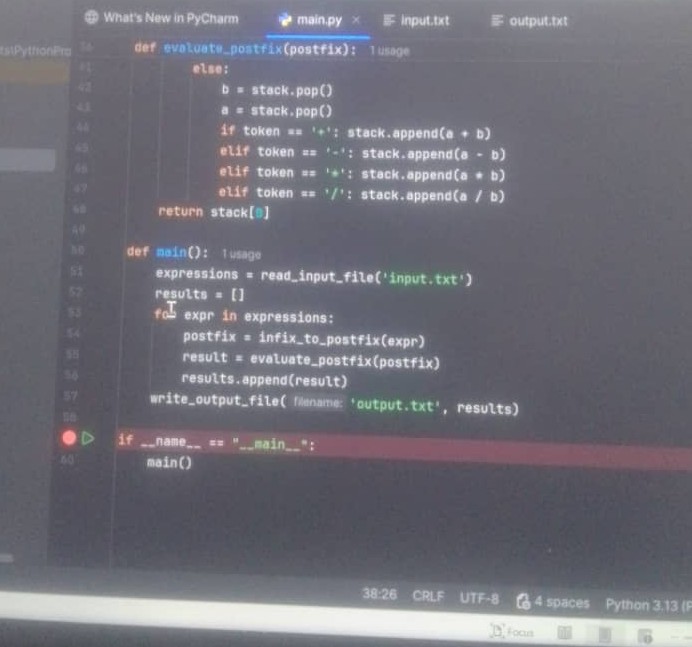
**Output.txt**



**PyCharm running the code**







**Conclusion**

This assignment demonstrates the practical application of stacks in parsing and evaluating mathematical expressions. It reinforces understanding of operator precedence, file handling, and modular programming in Python

**References**

• Python Documentation: https://docs.python.org/3/

• GeeksforGeeks: Infix to Postfix Conversion

• Stack Overflow discussions on stack evaluation