CHALLENGE 1:
ARITHMETIC EXPRESSION CALCULATION

1 Preliminaries

1.1 Research

While we use infix expressions in our daily lives, computers have trouble understanding this format because they need to keep in mind rules of operator precedence and also brackets. Prefix and Postfix expressions are easier for a computer to understand and evaluate [*]. Below are some examples of Infix, Prefix, and Postfix expressions.

Infix Expression	Prefix Expression	Postfix Expression
A + B	+ A B	A B +
A + B * C	+ A * B C	A B C * +
(A + B) * C	* + A B C	A B + C *
A + B * C + D	+ + A * B C D	A B C * + D +
(A + B) * (C + D)	* + A B + C D	A B + C D + *

Table 1: Examples of Infix, Prefix, and Postfix expressions

Explain the algorithms, step-by-step example, for the following algorithms:

* Converting an infix expression to a prefix expression.
* Converting an infix expression to a postfix expression.
* Converting a prefix expression to a postfix expression.

[*] geeksforgeeks

1.2 Programming

- You are requested to build a program for validating, converting and calculating the arithmetic infix expressions. Please choose one of the following options for further requirements.
  - **Integer option**: Input expressions contain only integers. *(8 points max)*
  - **Floating-point option**: Input expressions contain both integers and floating-point numbers. *(10 points max)*

- **Input**: the file *"input.txt"* consists of infix expressions, each located on a single line.

  Example:
  - (1 + 1) ^ 3
  - 3 * 2 + 1.5
  - 4 4 3 + - *(error expression)*
  - (3.72 + 5.16) / 2.22

  There might be some error expressions in the input file. However, the expressions must follow the below format:
  - Operands and operators from these expressions must be separated by one single space " ".
  - Legal operators: + (addition) , − (subtraction) , ∗ (multiplication), / (division) , ^ (power).
  - Legal brackets: ( ), [ ], { }
  - For the **Floating-point option**, real numbers have at most 2 decimal digits.

- Your program must be built into an execution *".exe"* file that runs within the following command line arguments:

```
A.exe InputPath N Action OutputPath
```

  in which:

  * `A.exe`: Your execution *".exe"* file.
  * `InputPath`: Path to the *"input.txt"* file.
  * `N`: an integer, represent the number of expressions from the *"input.txt"* file.
  * `Action`:
    - `-c`: calculate the identified the expressions.
    - `-t`: convert the identified the expressions into postfix expressions.
  * `OutputPath`: Path to the *"output.txt"* file, which will be mentioned below.

- **Output**: the file *"output.txt"*. Content of this file depends on the action chosen from the command line argument.

  * `-c`:
    - Calculated value of the expressions from input file. Real numbers result in 2 decimal digits. Error expressions result in **E**.
    - Each value locates on a single line, corresponding to its expression line from the input file.

  Examples:

Input	**Output**
(1 + 1) ˆ 3	8
3 * 2 + 1.5	7.5
4 4 3 + -	E
(3.72 + 5.16) / 2.22	4

∗ `-t`:
- Converted postfix expressions of the expressions from input file. Error expressions result in **E**.
- Converted postfix expressions format must follow the input expression format above.
- Each result postfix expression locates on one single line, corresponding to its infix expression line from the input file.

Examples:

Input	**Output**
(1 + 1) ˆ 3	1 1 + 3 ˆ
3 * 2 + 1.5	3 2 * 1.5 +
4 4 3 + -	E
(3.72 + 5.16) / 2.22	3.72 1.56 + 2.22 /

2 Group registration and Submission regulations

2.1 Group Registration

- This challenge requires a group of 3 - 4 students. Group members for each challenge must be different (i.e, Any pairs of students are at max ONE same group).
- Group ID is generated by concatenating the last 2 digits of each member's Student ID in ascending order.

  Example:

  – Given the student codes: *19127666 - 19127888 - 19127991 - 19127999.*

  → **Generated ID**: *66889199.*

  – Given the student codes: *19127667 - 19127889 - 19127990*

  → **Generated ID**: *678990.*

- Group registration will be provided within the attached link. **Each group's member** must fill in the registration form.

  Note: Group registration and file submission time should not be different more than 15$mins$.

2.2 Submission regulations

- ONLY 10 first submissions are accepted.
- The submission file must be in the following format: [**Group_ID.rar**] or [**Group_ID.zip**], is the compression of the [**Group_ID**] folder. This folder contains:
  - The report file must be presented as a document [**Group_ID.pdf**] or as a slideshow [**Group_ID.pptx**]. This file presented research answers from **1.1** and the information of code fragment (data structures, algorithms, functions) from **1.2**.
    - If your submission is a slideshow, there must be explanation in the *Note* part of each slide.
    - Information (Names, Student IDs) must be declared clearly on the first page (or first slide) of your report. Your working progress (Which option did you choose? How much work have you completed?) should be demonstrated on this page, too.
    - The report file should be **structured, logical, clear** and **coherent**. The length of the submission should not exceed 15 pages for the document file, and 30 pages for the presentation slide.
    - All links and books related to your submission must be mentioned.
  - The programming file must be a single file [**Group_ID.cpp**]. The code fragment must be clear, logical and commented.
- Submission with wrong regulation will result in a "0" (zero).
- Plagiarism and Cheating will result in a "0" (zero) for the entire course and will be subject to appropriate referral to the Management Board of the CLC program for further action.