**EXPRESSION EVALUATOR REPORT DOC**

NAME: B SHANTANU

YEAR: IV

COLLEGE: CVR COLLGE OF ENGINEERING

**DATA STRUCTURE:**

-Two STACKS

* Character Stack to convert given Infix to Postfix Expression
* Integer Stack for evaluating the obtained Postfix Expression

**ALGORITHM:**

-Converting from Infix to Postfix

* Use of Character Stack
* Push and Pop operators in the Stack
* Append Numbers into a String
* Pushing and Popping of operators are based on precedence of occurrence of operators
* Operators Supported – ‘+’, ‘-’, ‘\*’, ‘/’, ‘\*\*’
* Even negative numbers are handled
* Multi digit numbers are also handled

-Evaluating Postfix Expression

* Numbers are separated by space so that multi digit numbers can be easily identified
* Use of Integer Stack to store Values
* Integers are pushed onto stack and popped when operator comes, integers are popped and result is evaluated and pushed back to stack
* Final result will be on top of the Stack

-Precedence

* Power has highest Precedence
* Division and Multiplication have medium Precedence
* Addition and Subtraction have lower Precedence

Stacks are better data structure because we can compare the top element of the stack with the current element about the precedence .Last In First Out plays a vital role in conversion as the recently added operators are removed first until precedence is matched.

**Features of My Code:**

* Handles Multi Digit Values
* Handles Negative Numbers
* Support of Power Function
* Supports reading of a file and getting infix Expression from that

**Extensions**:

* Can be extended to handle errors like division by zero
* Can be extended to support decimal values