Ex. 1: Write a simple calculator program in C/C++/JAVA

Features of the calculator:

- Take input from standard input.
- On each line an arithmetic expression can be given in the standard format and the calculator
- must print the o/p after that.
- The calculator should exit, when the user enters the Ctrl^D (eof) character.
- Supported operators: +, -, *, /, ^, and ().

Procedure for simple calculator:

1. Read a line of input
2. If input is Ctrl^D terminate the program
3. If the expression is not balanced for parenthesis report error and go to step 1
4. Convert the expression into post-fix notation
5. Scan through the post-fix expression and push the operands into the stack in the order they appear
6. When any operator is encountered pop the top two stack elements (operands) and execute the operation
7. Push the result on to the stack
8. Print the result and got to step 1

Procedure for post-fix conversion:

1. Scan the expression from left to right
2. If the scanned character is operand place it in postfix expression
3. If the scanned character is an operand

a. If the precedence and associativity of the scanned operator is larger than the one in the stack then push it onto the stack.

b. Else pop all the operators from the stack, which are greater than or equal to in precedence and place it in postfix expression and push the scanned operator into the stack. Popping should be stopped if a parenthesis is encountered.

4. If the scanned character is a '(' push it into the stack
5. If the scanned character is a ')', pop the stack and output it until a '(' is encountered, and discard both the parenthesis.
6. Repeat steps 2-5 until the infix expression is scanned.
7. Once the scanning is over, pop the stack and add the operators in the postfix expression until it is not empty.
8. Return the postfix expression.

Procedure for Balancing Symbols:

1. Traverse the expression from left to right.

a. If the current character is a starting bracket, then push it to stack

b. If the current character is a closing bracket, then pop from stack and if the poppedcharacter is not matching starting bracket, return “not balanced”.

2. If there is some starting bracket left in stack then return “not balanced”

3. Return balanced

Example run of the program:

Input: 2 + 3 * 5

Output: 17

Input 2: (2 + 3) * 5

Output 2: 25

Input 3: 30 / 5 / 2

Output 3: 3

Input 4: 2 ^ 3 ^ 2

Output 4: 64

Input 5: (2 ^ 3) ^ 2

Output 5: 36