Lab Assignment: Infix to Postfix Conversion

Objective

To convert an infix expression to its corresponding postfix expression using stack data structure. This includes handling of operators with different precedence and associativity, including right-associative operators like exponentiation (^).

Problem Statement

Write a C program to convert a given infix expression to postfix notation. The infix expression include operands with single characters , operators (+, -, *, /, ^), and parentheses.

Function Signature

void infixToPostfix(char* infix, char* postfix);

Algorithm

1. Initialize an empty stack for operators.

2. Traverse the infix expression from left to right.

3. For each character in the expression:

   a. If it is an operand, add it to the postfix expression.

   b. If it is an opening parenthesis '(', push it onto the stack.

   c. If it is a closing parenthesis ')', pop from the stack to the postfix expression until '(' is encountered.

   d. If it is an operator, then:

      i. While the stack is not empty and the precedence of the current operator is less than or equal to the precedence of the operator at the top of the stack,

         or the current operator is right-associative and has the same precedence as the top of the stack, pop from the stack to the postfix expression.

      ii. Push the current operator onto the stack.

4. After the expression is fully traversed, pop all remaining operators from the stack to the postfix expression.

Notes

• Operators '+', '-', '*', and '/' are left-associative.

• The operator '^' (exponentiation) is right-associative.

• Proper handling of precedence and associativity is necessary for correct output.

Sample Inputs and Expected Outputs

Input: a + b * c

Output: abc*+

Input: a * (b + c * d)

Output: abcd*+*

Input: a ^ b ^ c

Output: abc^^