**Algorithm to Convert Infix to Postfix Expression Using Stack**

1. Initialize the Stack.
2. Scan the operator from left to right in the infix expression.
3. If the leftmost character is an operand, set it as the current output to the Postfix string.
4. And if the scanned character is the operator and the Stack is empty or contains the '(', ')' symbol, push the operator into the Stack.
5. If the scanned operator has higher precedence than the existing **precedence** operator in the Stack or if the Stack is empty, put it on the Stack.
6. If the scanned operator has lower precedence than the existing operator in the Stack, pop all the Stack operators. After that, push the scanned operator into the Stack.
7. If the scanned character is a left bracket '(', push it into the Stack.
8. If we encountered right bracket ')', pop the Stack and print all output string character until '(' is encountered and discard both the bracket.
9. Repeat all steps from 2 to 8 until the infix expression is scanned.
10. Print the Stack output.
11. Pop and output all characters, including the operator, from the Stack until it is not empty.

**Algorithm to Convert Infix to Prefix Expression Using Stack**

1.First, reverse the infix expression given in the problem.

2.Scan the expression from left to right.

3.Whenever the operands arrive, print them.

4.If the operator arrives and the stack is found to be empty, then simply push the operator into the stack.

5.If the incoming operator has higher precedence than the TOP of the stack, push the incoming operator into the stack.

6.If the incoming operator has the same precedence with a TOP of the stack, push the incoming operator into the stack.

7.If the incoming operator has lower precedence than the TOP of the stack, pop, and print the top of the stack. Test the incoming operator against the top of the stack again and pop the operator from the stack till it finds the operator of a lower precedence or same precedence.

8.If the incoming operator has the same precedence with the top of the stack and the incoming operator is ^, then pop the top of the stack till the condition is true. If the condition is not true, push the ^ operator.

9.When we reach the end of the expression, pop, and print all the operators from the top of the stack.

10.If the operator is ')', then push it into the stack.

11.If the operator is '(', then pop all the operators from the stack till it finds ) opening bracket in the stack.

12.If the top of the stack is ')', push the operator on the stack.

13.At the end, reverse the output.