Assignment Number: 8

Problem statement:

Expression conversion: Write a menu driven C++ program for expression conversion and evaluation

1. infix to prefix

2. prefix to postfix

3. prefix to infix

4. postfix to infix

5. postfix to prefix

6. infix to postfix

Objectives:

To know the applications of stack

To know the advantages of polish notations

Theory:

Applications of stack:

1. Converting infix expression to prefix and postfix form
2. Evaluating the postfix expression
3. Checking well-formed (nested) parenthesis
4. Reversing the string
5. Processing function calls
6. Parsing of computer programs
7. Simulating recursion
8. In computations such as decimal to binary conversion
9. In backtracking algorithms

Polish Notation:

Polish Notation is a way of expressing arithmetic expressions that avoids the use of brackets to define priorities for evaluation of operators. Polish Notation was devised by the Polish philosopher and mathematician Jan Łukasiewicz for use in symbolic logic. In his notation, the operators preceded their arguments, so that the Infix Notation expression

(3 + 5) * (7 - 2) (Infix)

would be written as

   * + 3 5 - 7 2 (Prefix)

The 'reversed' form, Reverse Polish Notation (RPN), has however been found more convenient from a computational point of view. In this notation the above expression would be

   3 5 + 7 2 - * (Postfix)

Infix expression: The expression of the form a op b. When an operator is in-between every pair of operands.
Postfix expression: The expression of the form a b op. When an operator is followed for every pair of operands.
Prefix expression: The expression of the form op a b

The compiler uses polish notations.

Algorithm:

For infix to postfix conversion:

Input: Infix expression

Output : **Postfix expression**

```
Step1: Scan expression E from left to right, character by character, till character is #
                        Ch= get_next_token(E)
Step2: While (ch!= #)
    If (ch = ')')then ch = pop()
        While (ch!= '(')
            Display ch
            Ch=pop()
        End while
    If (ch = operand) display the same
    If (ch = operator) then
        If (ICP > ISP) then push ch
        Else
            While (ICP<=ISP)
                Pop the operator and display it
            End while
        Ch = get_next_token(E)
    End while
Step 3: If (ch = #) then while (! Emptystack()) pop and display
Step 4: Stop
```

Time complexity:

For infix to postfix conversion: _________

For infix to prefix conversion: _________

For evaluating postfix expression: _________

Test cases

1. Evaluate the postfix expression AB*C+ , where A=4, B=5, C=6
2. Evaluate the postfix expression ABC^^ , where A=2, B=3, C=4
3. Convert infix expression A*B+C into polish notation
4. Convert infix expression A^B*C-C+D/A/(E+F) into polish notation

Conclusion: Thus we have successfully implemented the stack for expression conversion.

Practice problem:

Write algorithm for following operations

1. infix to prefix 2. prefix to postfix 3. prefix to infix

4. postfix to infix 5. postfix to prefix 6. Evaluation of Expression