

SE(ECE)	Data Structures and Algorithms	
Experiment No: 7	Evaluate postfix expression (input will be postfix expression)	Page: 1/3

### Aim

To implement a program in C++ for evaluation of a postfix (Reverse Polish Notation) expression using a stack.

### Objectives

1. To understand the concept of postfix expressions (Reverse Polish Notation).
2. To learn how stack is used for expression evaluation.
3. To implement a program in C++ that evaluates postfix expressions.
4. To apply expression evaluation concepts in compiler design and calculators.

### Theory

#### Postfix Expression

- A postfix expression (Reverse Polish Notation) is a way of writing expressions without parentheses.
- Operators are written after their operands.
  - Example: Infix  $\rightarrow A + B * C$   
Postfix  $\rightarrow A B C * +$

#### Algorithm: Evaluation of Postfix Expression

1. Initialize an empty stack.
2. Scan the postfix expression from left to right.
3. If the scanned symbol is an operand, push it onto the stack.
4. If the scanned symbol is an operator (+, -, \*, /, ^):
  - Pop two operands from the stack.
  - Apply the operator.
  - Push the result back onto the stack.
5. After the entire expression is scanned, the final result will be on the stack top.

### Applications

1. Compilers - for evaluating arithmetic expressions.
2. Calculators - postfix expressions simplify evaluation.
3. Expression parsing in programming languages.
4. Stack-based machines use postfix for instruction execution.

<b>SE(ECE)</b>	<b>Data Structures and Algorithms</b>	
<b>Experiment No: 7</b>	<b>Evaluate postfix expression (input will be postfix expression)</b>	<b>Page: 2/3</b>

### References

1. Yedidyah Langsam, Moshe J Augenstein, Aaron M Tenenbaum – Data structures using C and C++ - PHI Publications ( 2nd Edition ).
2. Ellis Horowitz, Sataraj Sahni- Fundamentals of Data Structures – Galgotia Books source.

### Questions

1. The postfix form of the expression  $(A + B) * (C * D - E) * F / G$  is?
  - a)  $AB + CD * E - FG / **$
  - b)  $AB + CD * E - F ** G /$
  - c)  $AB + CD * E - * F * G /$
  - d)  $AB + CDE * - * F * G /$
2. The data structure required to check whether an expression contains balanced parenthesis is?
  - a) Stack
  - b) Queue
  - c) Array
  - d) Tree
3. The postfix form of  $A * B + C / D$  is?
  - a)  $* AB / CD +$
  - b)  $AB * CD / +$
  - c)  $A * BC + / D$
  - d)  $ABCD + / *$
4. Which data structure is needed to convert infix notation to postfix notation?
  - a) Branch
  - b) Tree
  - c) Queue
  - d) Stack
5. The prefix form of  $A - B / (C * D \wedge E)$  is?
  - a)  $- / * \wedge ACBDE$
  - b)  $- ABCD * \wedge DE$
  - c)  $- A / B * C \wedge DE$
  - d)  $- A / BC * \wedge DE$
6. The prefix form of an infix expression  $p + q - r * t$  is?
  - a)  $+ pq - * rt$
  - b)  $- + pqr * t$
  - c)  $- + pq * rt$
  - d)  $- + * pqrt$

<b>SE(ECE)</b>	<b>Data Structures and Algorithms</b>	
<b>Experiment No: 7</b>	<b>Evaluate postfix expression (input will be postfix expression)</b>	<b>Page: 3/3</b>

- 7. The result of evaluating the postfix expression 5, 4, 6, +, \*, 4, 9, 3, /, +, \* is?**
- 600
  - 350
  - 650
  - 588
- 8. Convert the following infix expressions into its equivalent postfix expressions  $(A + B \wedge D)/(E - F) + G$**
- $(A B D \wedge + E F - / G +)$
  - $(A B D + \wedge E F - / G +)$
  - $(A B D \wedge + E F / - G +)$
  - None
- 9. The type of expression in which operator succeeds its operands is?**
- Infix Expression
  - pre fix Expression
  - postfix Expression
  - None
- 10. Which of the following application generally use a stack?**
- Parenthesis balancing program
  - Syntax analyzer in compiler
  - Keeping track of local variables at run time
  - All of the above
- 11. Consider the usual implementation of parentheses balancing program using stack. What is the maximum number of parentheses that will appear on stack at any instance of time during the analysis of  $((())(())?)$**
- 1
  - 2
  - 3
  - 4

## Conclusion

-----

-----

A	C	O	T	Sign.