CS 251
Intermediate Programming
Writing a Postfix Calculator

Brooke Chenoweth

University of New Mexico

Spring 2025

Stack Operations

A stack is a LIFO structure.

- isEmpty – Is the stack empty?
- push – add an item to the stack
- pop – remove item from stack
- peek – look at item on top of stack without changing it

Infix vs PostFix

- Infix notation – operator is between operands
  - X + Y
  - A * B + C / D
  - A * (B + C) / D
- Postfix notation – operator is after operands
  - X Y +
  - A B * C D / +
  - A B C + * D /

Postfix calculator

1. Start with empty stack
2. Scan expression
3. While we still have tokens:
   - If current token is an operand (a number), push it onto the stack
   - If current token is an operator:
     - 3.1 Pop operands off the stack
     - 3.2 Evaluate operator with operands
     - 3.3 Push resulting value onto the stack.
4. Remaining element in stack is final result of postfix expression.

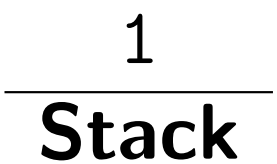
Examples

- 1 2 3 * + 4 -
- 5 1 2 + 4 * + 3 -
- 1 2 + 3 * 6 + 2 3 + /

Example

Postfix Expression: `1 2 * 3 4 * + 5 *`

- `1`



Example

Postfix Expression: `1 2 * 3 4 * + 5 *`

- 1
- 2

2
1
Stack

Example

Postfix Expression: `1 2 * 3 4 * + 5 *`

- 1
- 2
- *

2

Stack

Example

Postfix Expression: `1 2 * 3 4 * + 5 *`

- 1
- 2
- *
- 3

3
2

Stack

Example

Postfix Expression: `1 2 * 3 4 * + 5 *`

- 1
- 2
- *
- 3
- 4

4
3
2

Stack

Example

Postfix Expression: `1 2 * 3 4 * + 5 *`

- 1
- 2
- *
- 3
- 4
- *

12
2

Stack

Example

Postfix Expression: `1 2 * 3 4 * + 5 *`

- 1
- 2
- *
- 3
- 4
- *
- +

14

Stack

Example

Postfix Expression: `1 2 * 3 4 * + 5 *`

- 1
- 2
- *
- 3
- 4
- *
- +
- 5

5
14

Stack

Example

Postfix Expression: `1 2 * 3 4 * + 5 *`

- 1
- 2
- *
- 3
- 4
- *
- +
- 5
- *

70

Stack