CS 251
Intermediate Programming
Postfix to Infix

Brooke Chenoweth

University of New Mexico

Spring 2025

Converting Postfix to Infix

1. Create a stack to hold intermediate infix expressions.
2. For each token in the postfix expression:
    - If token is an operand, push onto stack
    - If operator, pop two subexpressions from stack, combine with operator to make new infix expression, push that onto stack
3. The item left on stack after postfix expression is processed is the final infix expression.

But what about parentheses?

- First subexpression is `1 + 2`
- Second subexpression is `3 * 4`
- What happens if I combine them with a multiplication operator?

But what about parentheses?

- First subexpression is `1 + 2`
- Second subexpression is `3 * 4`
- What happens if I combine them with a multiplication operator?
- Naive answer is `1 + 2 * 3 * 4`, but that's not what we wanted!

But what about parentheses?

- First subexpression is `1 + 2`
- Second subexpression is `3 * 4`
- What happens if I combine them with a multiplication operator?
- Naive answer is `1 + 2 * 3 * 4`, but that's not what we wanted!
- Could add parentheses around every subexpression, but that's overkill.

Adding parentheses only when needed

- Compare subexpression operator precedence with current operator
- If subexpression operator has lower precedence than current operator, add parentheses to subexpression before combining.
- `+` has lower precedence than `*`, so previous example would result in `(1 + 2) * 3 * 4`

Example

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

- `1`

1

Stack

Example

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

- `1`
- `2`

```
 2
 1
```

Stack

Example

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

- `1`
- `2`
- `*`

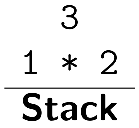
`1 * 2`

Stack

Example

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

- `1`
- `2`
- `*`
- `3`



Example

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

- 1
- 2
- *
- 3
- 4

```
  4
  3
1 * 2
-----
Stack
```

Example

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

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

```
3 * 4
1 * 2
```

Stack

Example

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

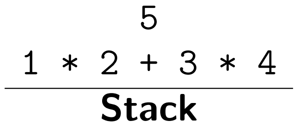
- `1`
- `2`
- `*`
- `3`
- `4`
- `*`
- `+`

`1 * 2 + 3 * 4`
Stack

Example

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

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



Example

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

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

`(1 * 2 + 3 * 4) * 5`

Stack

How can we implement in Java?

What do we need?

- Need to break up postfix expression into tokens.
- Need a stack of intermediate expressions.
- Need to know operator precedence.