CS 211

Project 5 – Infix Expression Evaluation

The Quick Version

More on Friday

Part 1 – Create you own Stack Class

- Need two stacks
  - one stack of integers for the operands
  - one stack of characters for the operators

- Do we need 2 Stack Classes?
  - Creating one Stack Class of integers works for this project since characters are a sub-type of integers (convert with type cast)

  - Attempting to use C++ templating is way too complicated!!
    - An INSANE idea for your first C++ Object program
    - Anyone using templating will be assumed to be copying and will need a meeting with Prof. Troy

Part 1 – Create your own Stack Class

- Requires implementation with a Dynamic Array
- Data Members:
  - pointer to a dynamic array
  - amount of space currently allocated
  - amount of space currently in use

- Methods
  - Constructor/Initialize
  - Reset (clear array)
  - IsEmpty()
  - Push()
  - Pop()
  - Top()

Part 2 – Evaluation Algorithm/Example

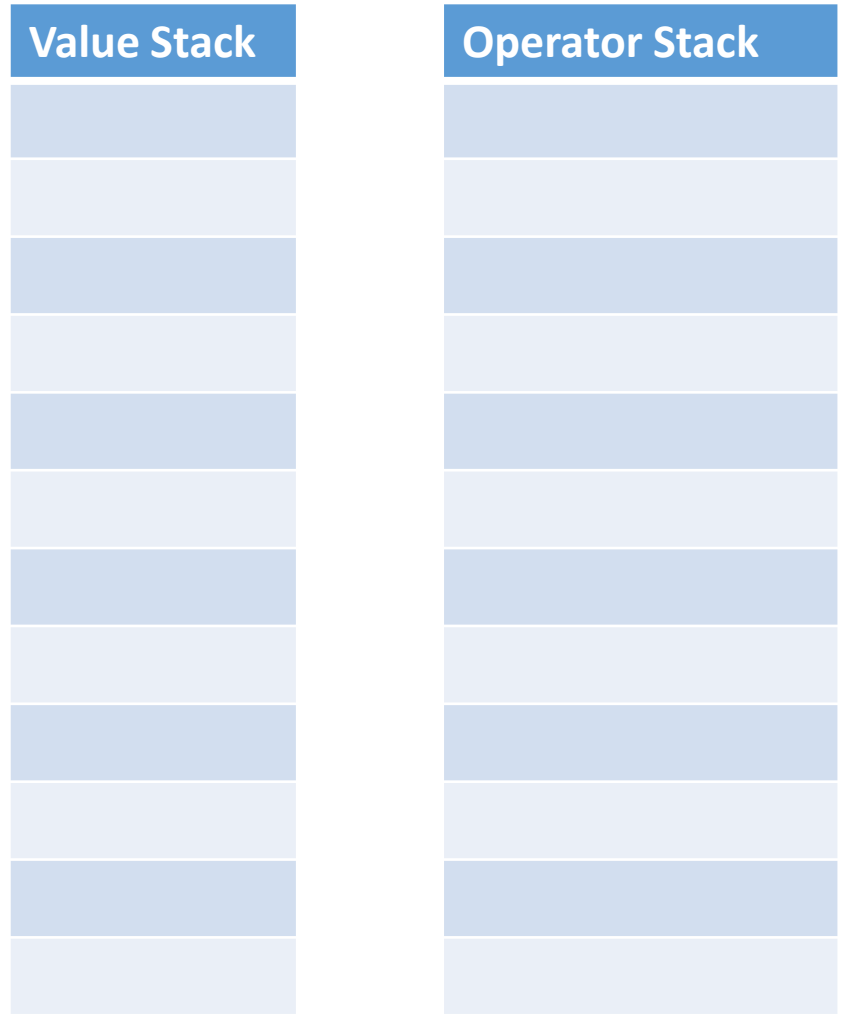
Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token:



Part 2 – Evaluation Algorithm/Example

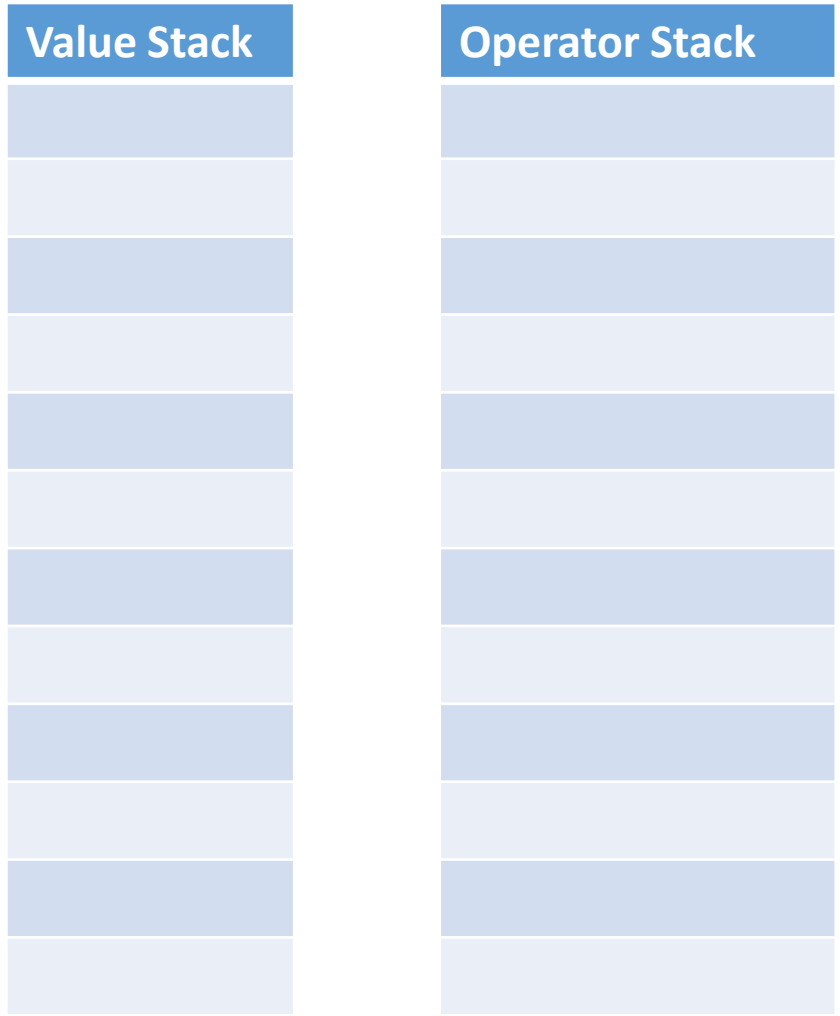
Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

```
3  *  4  +  5    EOLN
^
```

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: VALUE 3



Part 2 – Evaluation Algorithm/Example

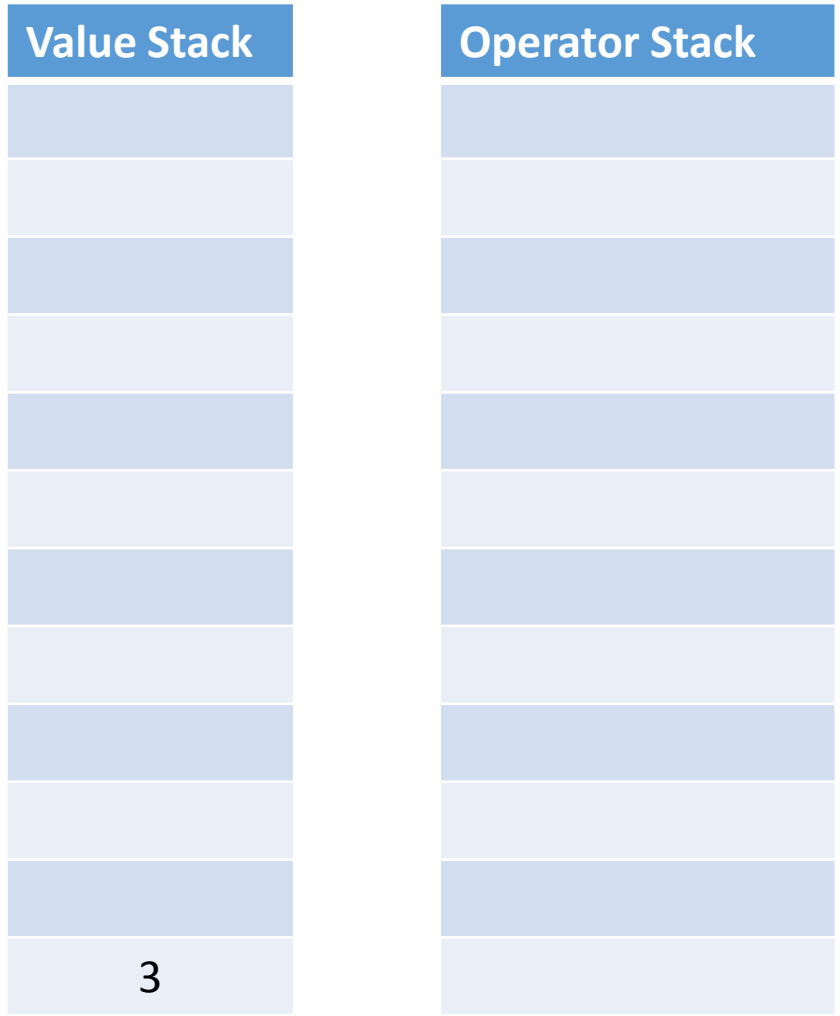
Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: VALUE 3
  - Push on Value Stack



Part 2 – Evaluation Algorithm/Example

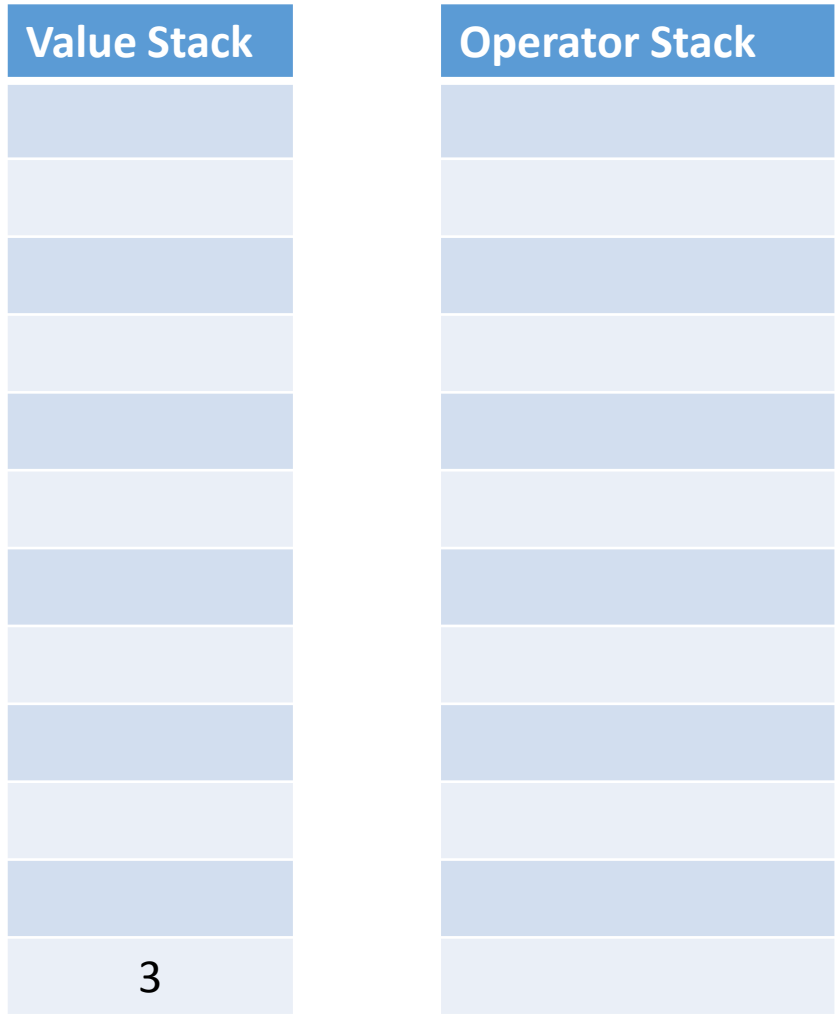
Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: OPERATOR *



Part 2 – Evaluation Algorithm/Example

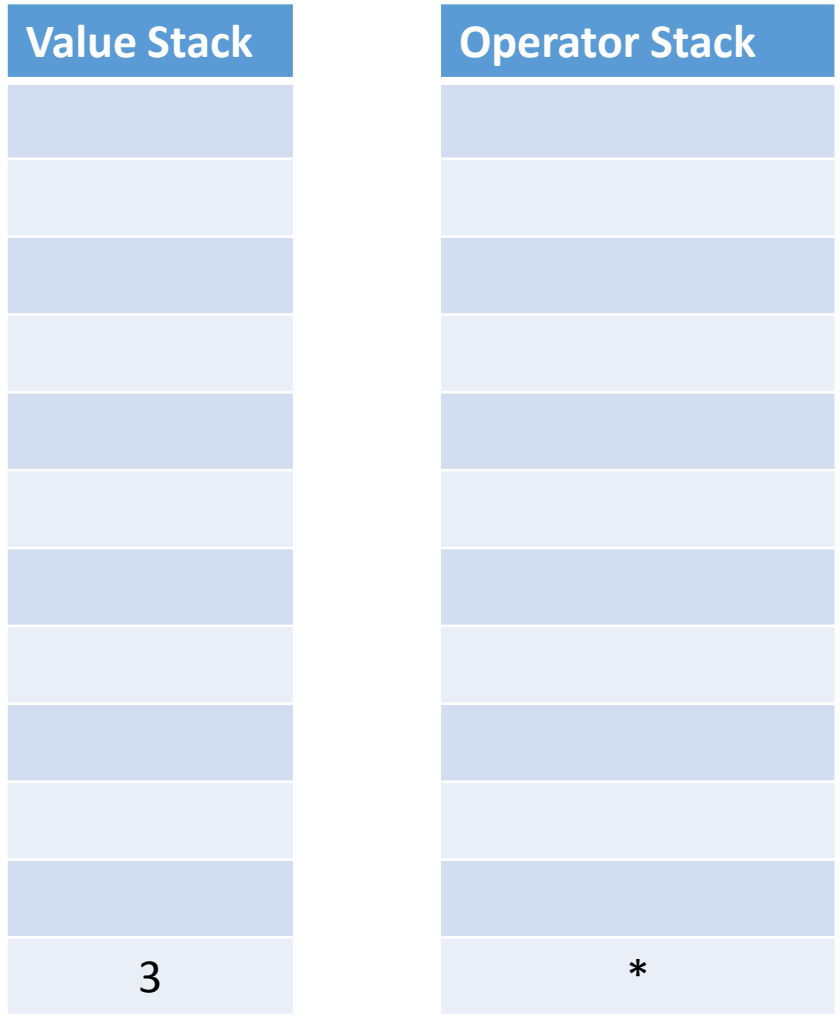
Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: OPERATOR *
  - Since Operator Stack is Empty
    Push on OP Stack



Part 2 – Evaluation Algorithm/Example

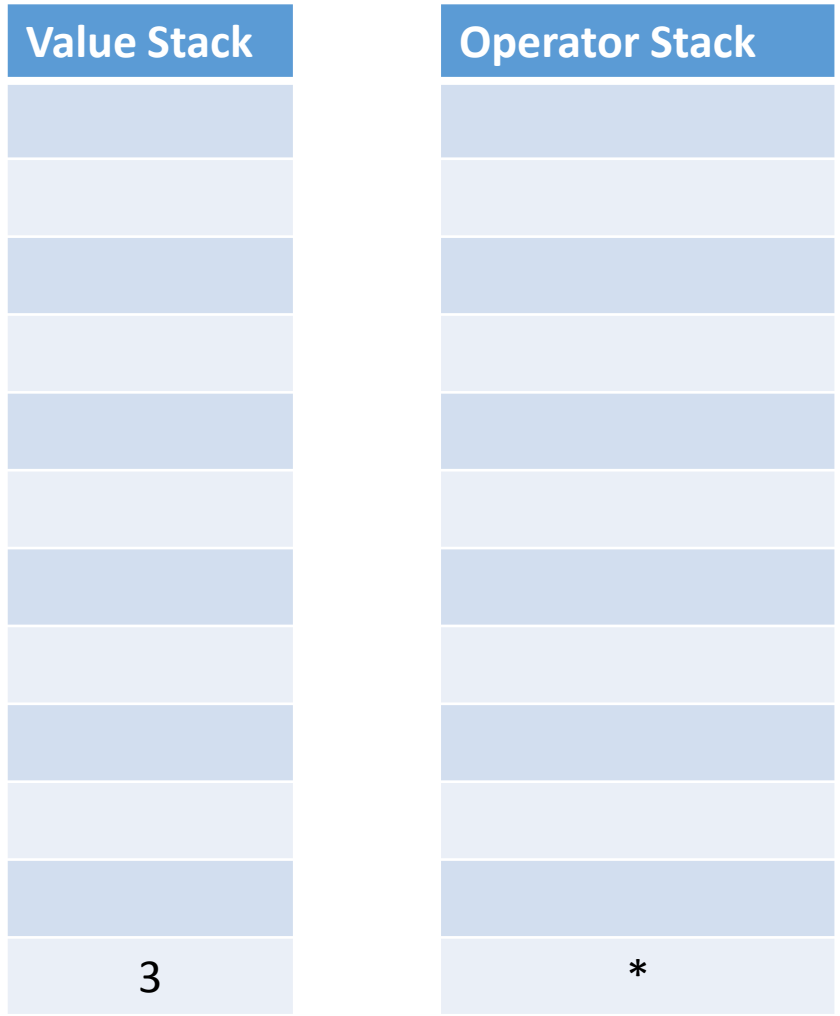
Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: VALUE 4



Part 2 – Evaluation Algorithm/Example

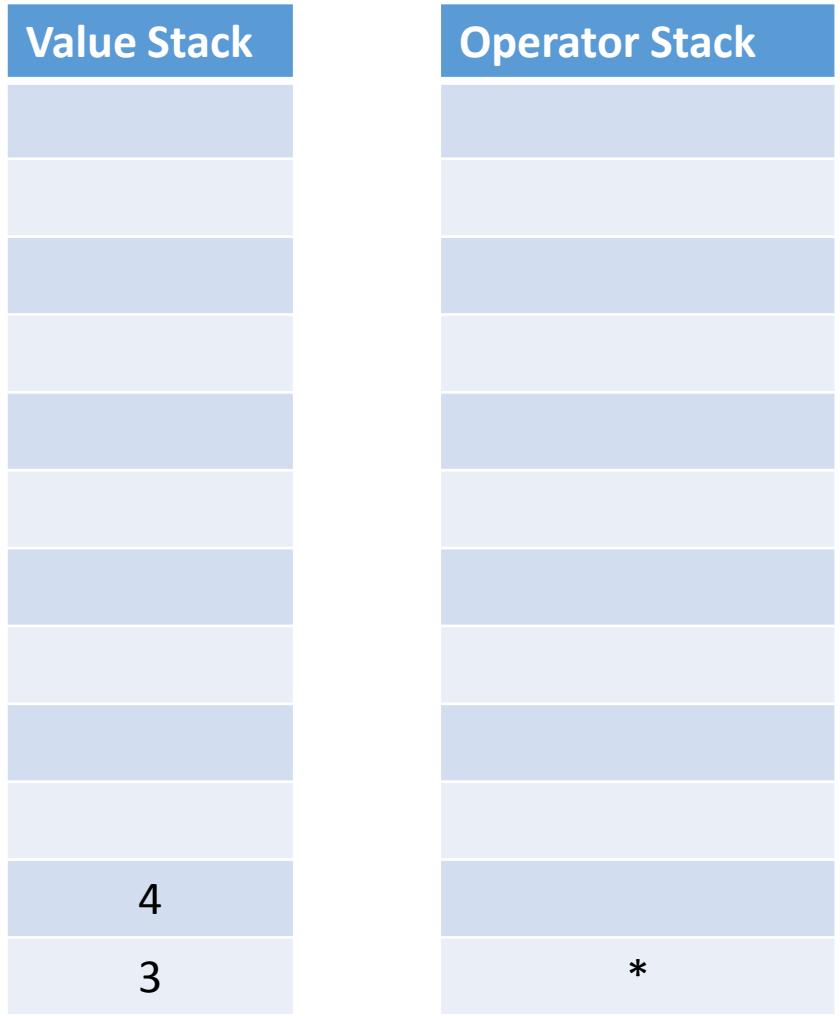
Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: VALUE 4
  - Push on VALUE Stack



Part 2 – Evaluation Algorithm/Example

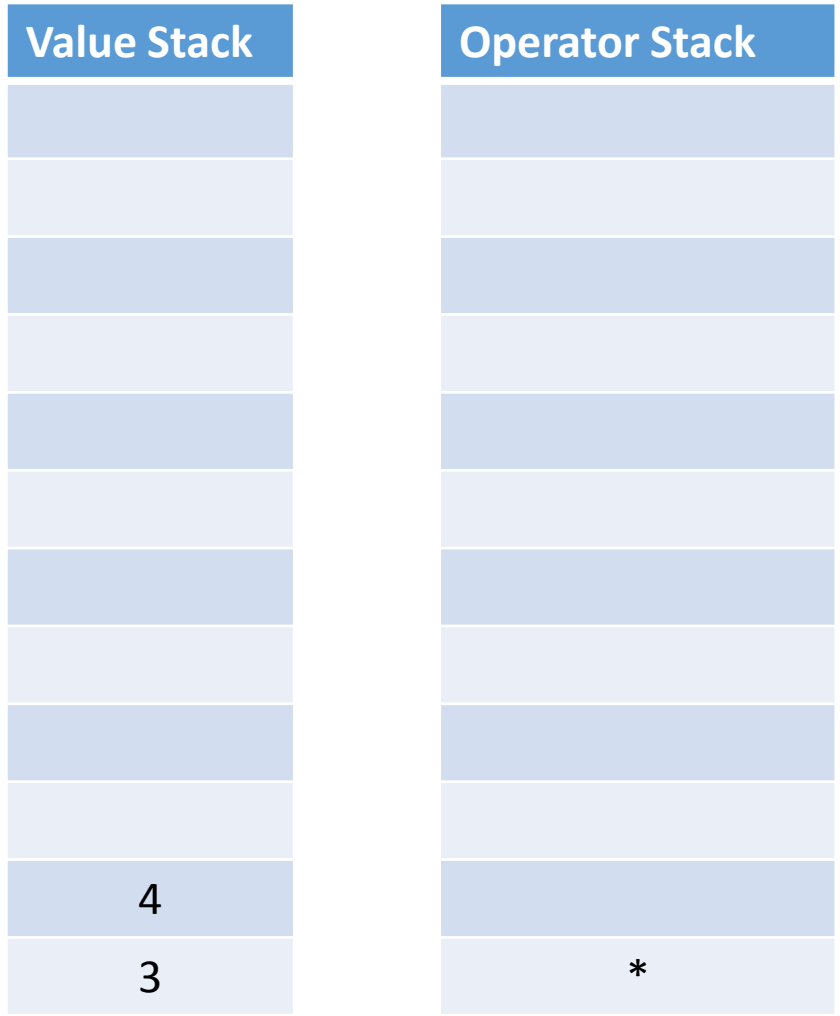
Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: OPERATOR +



Part 2 – Evaluation Algorithm/Example

Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

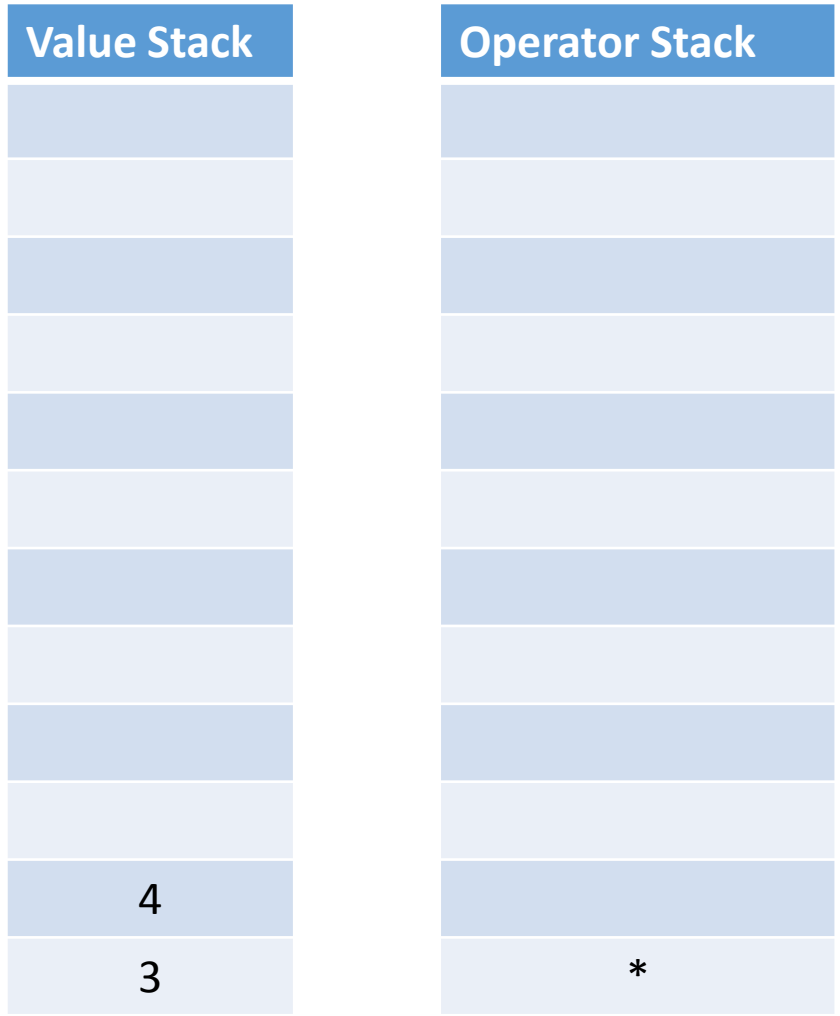
- Current Token: OPERATOR +
  - Since Top OP has higher precedence
    PopAndEval( ) then …

Value Stack	Operator Stack
4	
3	*

Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = Val1 Op Val2

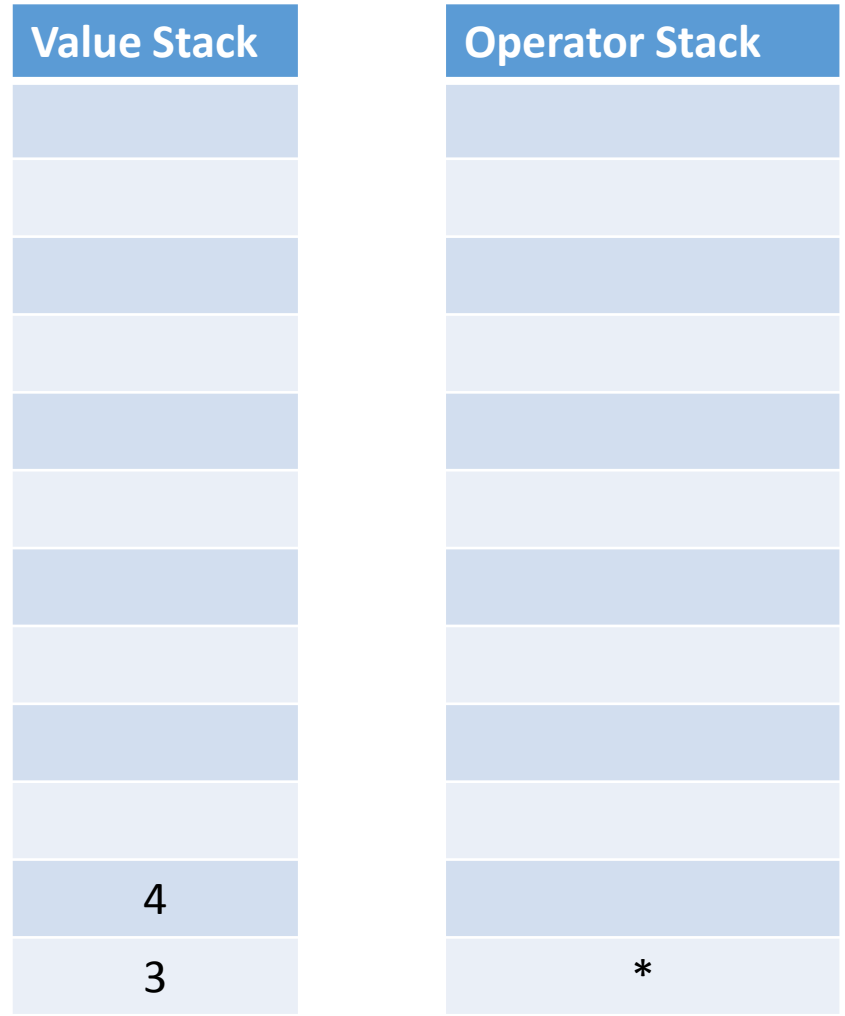


Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = Val1 * Val2

- Step 1: Op = Top of OP Stack

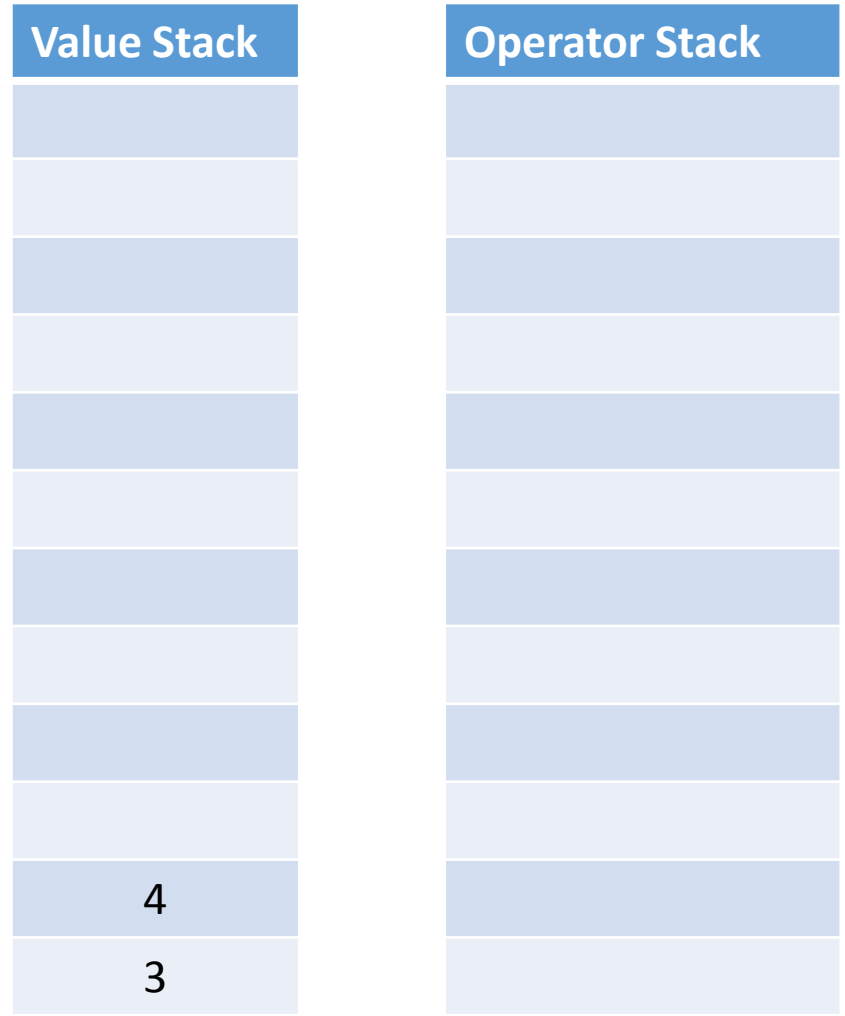


Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = Val1 * Val2

- Step 1: Op = Top of OP Stack
- Step 2: Pop OP Stack

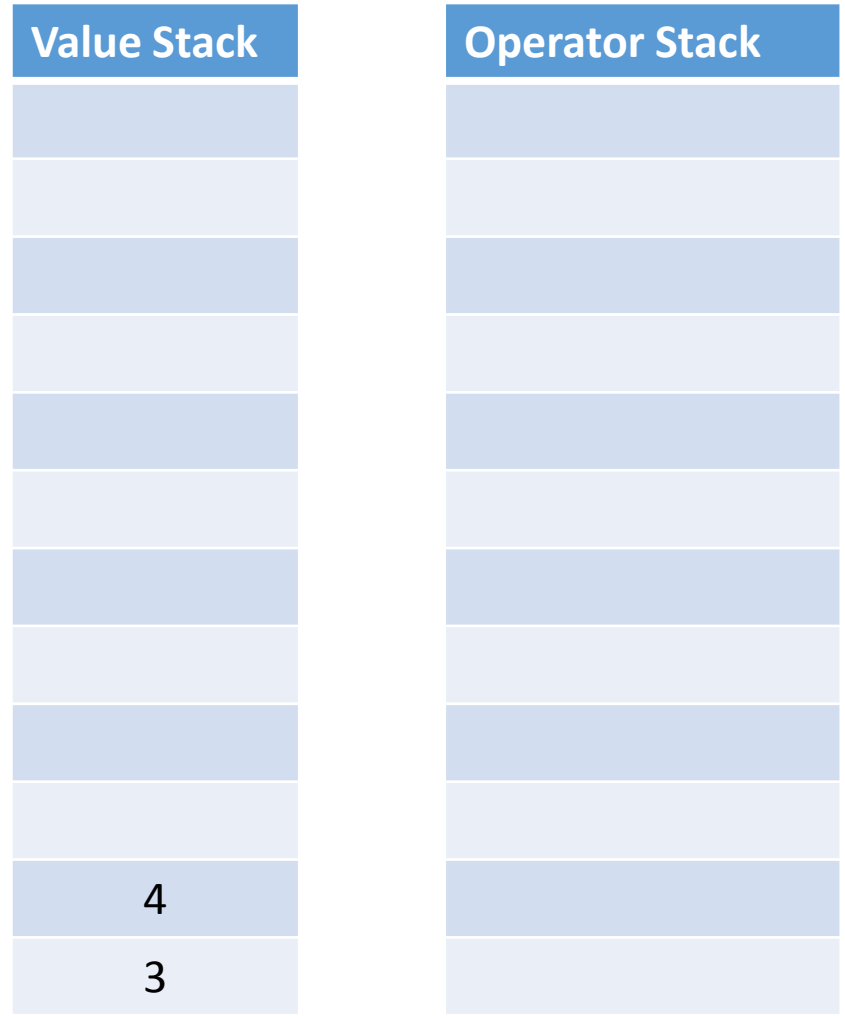


Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = Val1 * 4

- Step 1: Op = Top of OP Stack
- Step 2: Pop OP Stack
- Step 3: Val2 = Top of Value Stack

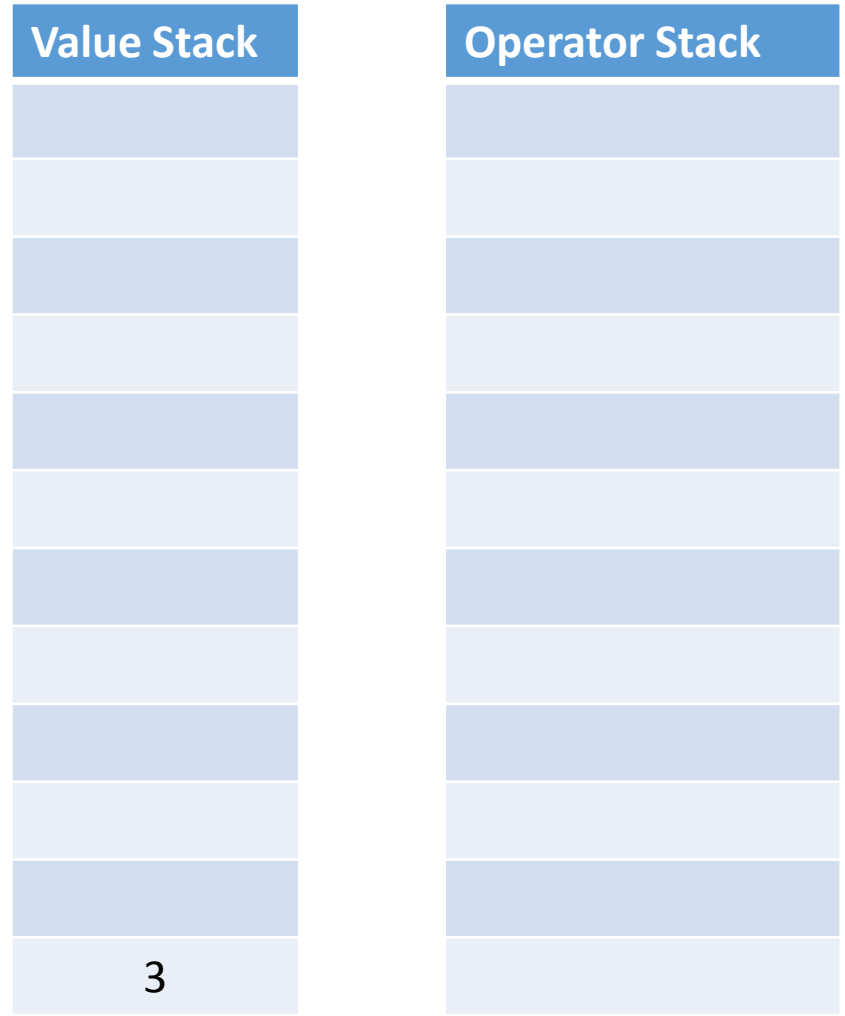


Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = Val1 * 4

- Step 1: Op = Top of OP Stack
- Step 2: Pop OP Stack
- Step 3: Val2 = Top of Value Stack
- Step 4: Pop Value Stack

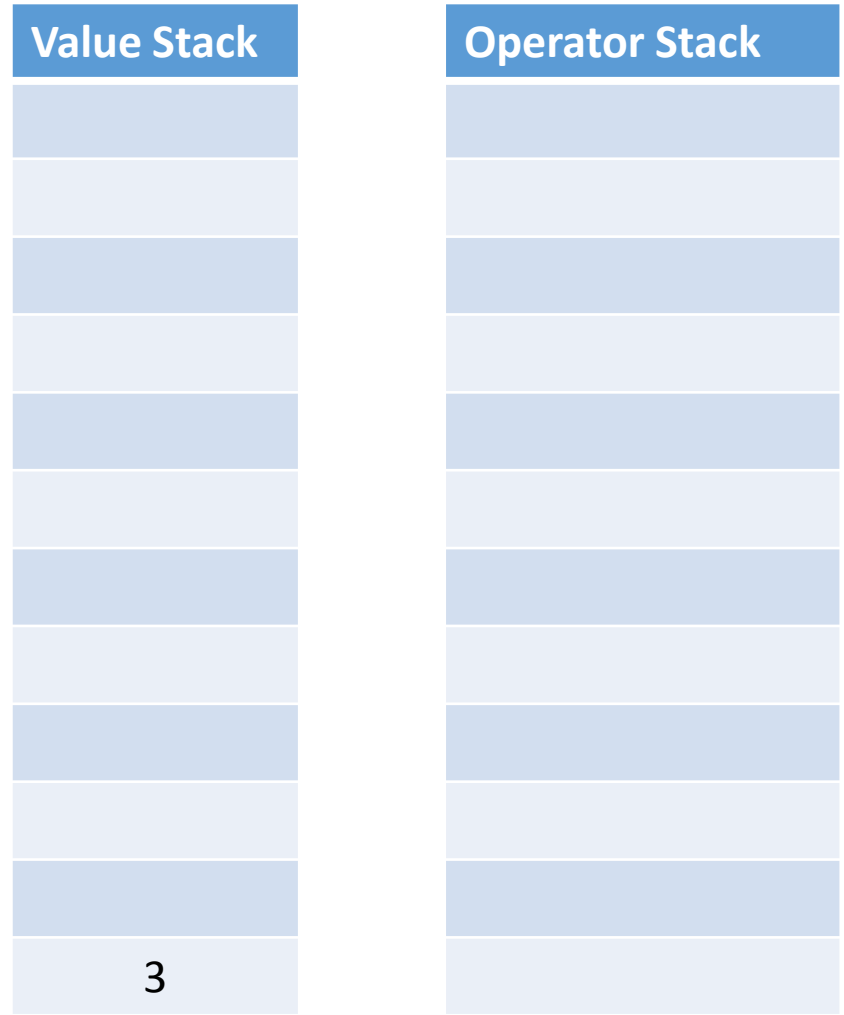


Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = 3 * 4

- Step 1: Op = Top of OP Stack
- Step 2: Pop OP Stack
- Step 3: Val2 = Top of Value Stack
- Step 4: Pop Value Stack
- Step 5: Val1 = Top of Value Stack

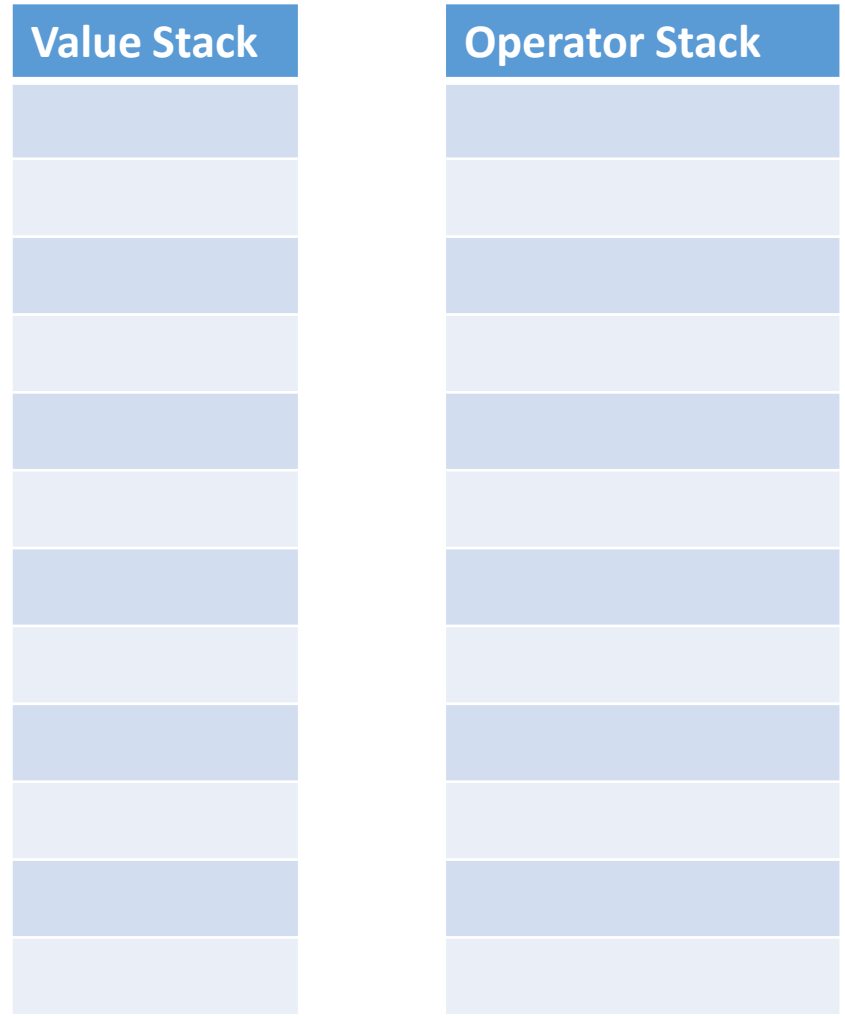


Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = 3 * 4

- Step 1: Op = Top of OP Stack
- Step 2: Pop OP Stack
- Step 3: Val2 = Top of Value Stack
- Step 4: Pop Value Stack
- Step 5: Val1 = Top of Value Stack
- Step 6: Pop Value Stack

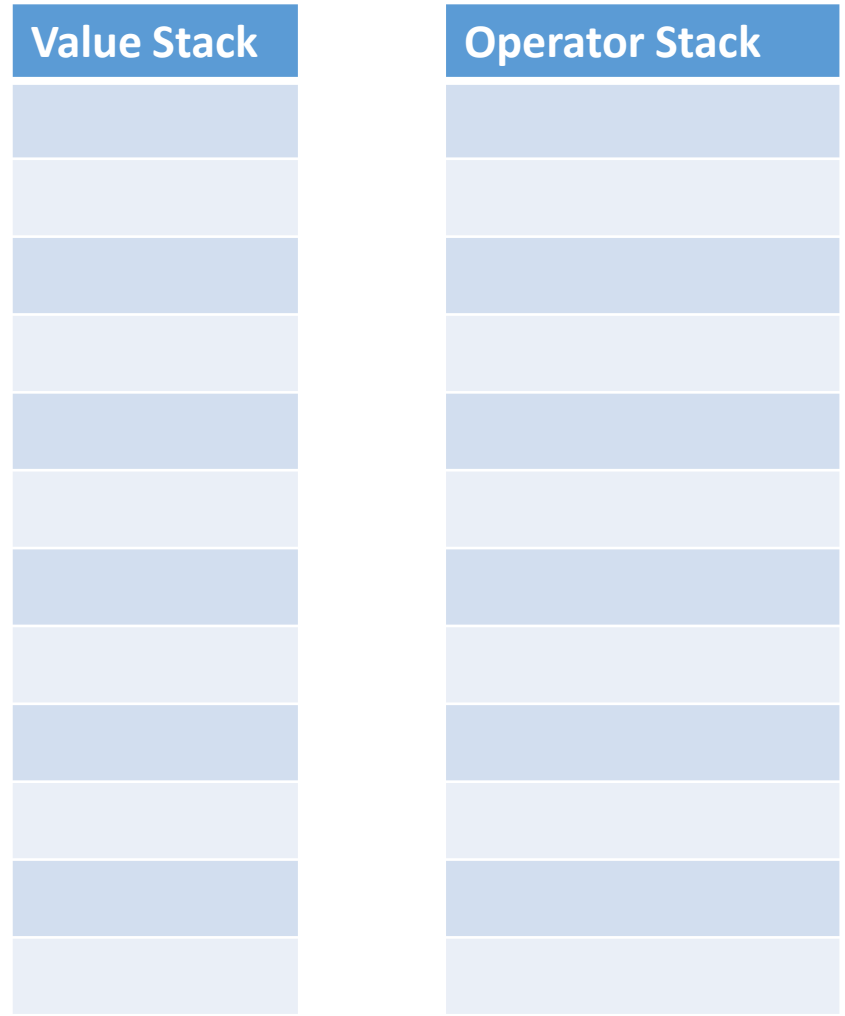


Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = 12

- Step 1: Op = Top of OP Stack
- Step 2: Pop OP Stack
- Step 3: Val2 = Top of Value Stack
- Step 4: Pop Value Stack
- Step 5: Val1 = Top of Value Stack
- Step 6: Pop Value Stack
- Step 7: Evaluate Result

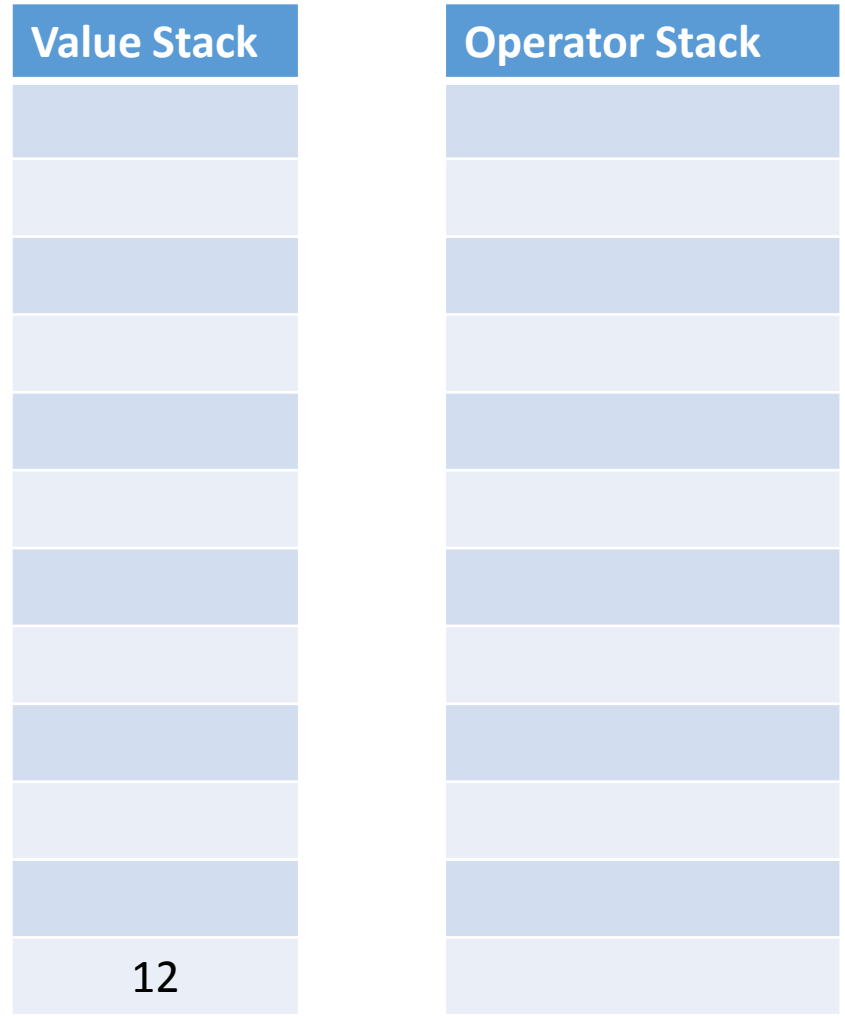


Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = 12

- Step 1: Op = Top of OP Stack
- Step 2: Pop OP Stack
- Step 3: Val2 = Top of Value Stack
- Step 4: Pop Value Stack
- Step 5: Val1 = Top of Value Stack
- Step 6: Pop Value Stack
- Step 7: Evaluate Result
- Step 8: Push Result on Value Stack



Part 2 – Evaluation Algorithm/Example

Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: OPERATOR +
  - Since Top OP has higher precedence
    PopAndEval( ) then Push on OP Stack

Value Stack	Operator Stack
12	+

Part 2 – Evaluation Algorithm/Example

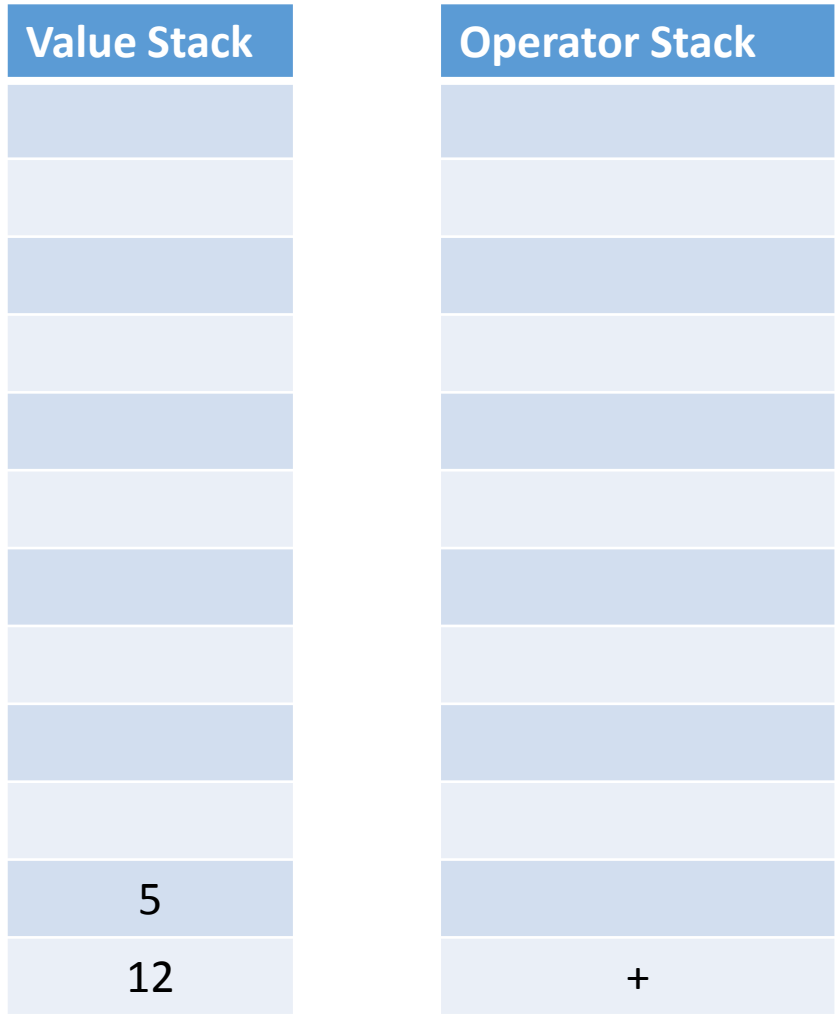
Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: Value 5
  - Push on Value Stack



Part 2 – Evaluation Algorithm/Example

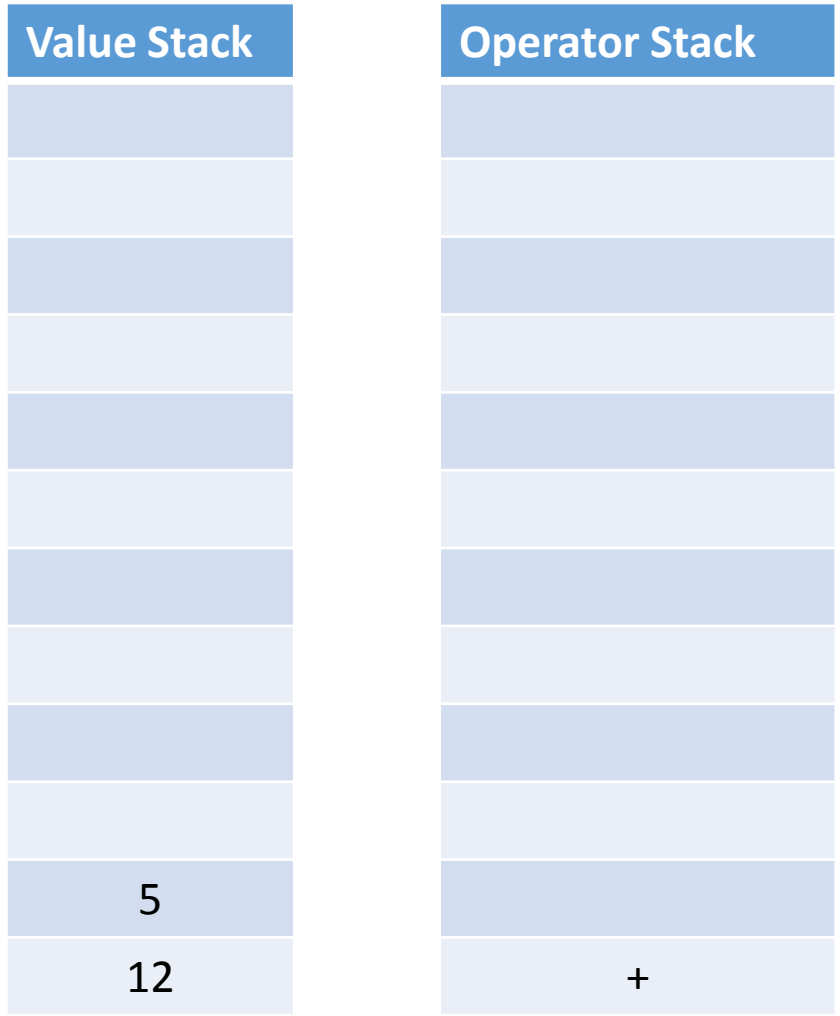
Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: EOLN
  - PopAndEval( ) until OP Stack is Empty



Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = 12 + 5

- Step 1: Op = Top of OP Stack
- Step 2: Pop OP Stack
- Step 3: Val2 = Top of Value Stack
- Step 4: Pop Value Stack
- Step 5: Val1 = Top of Value Stack
- Step 6: Pop Value Stack
- Step 7: Evaluate Result
- Step 8: Push Result on Value Stack

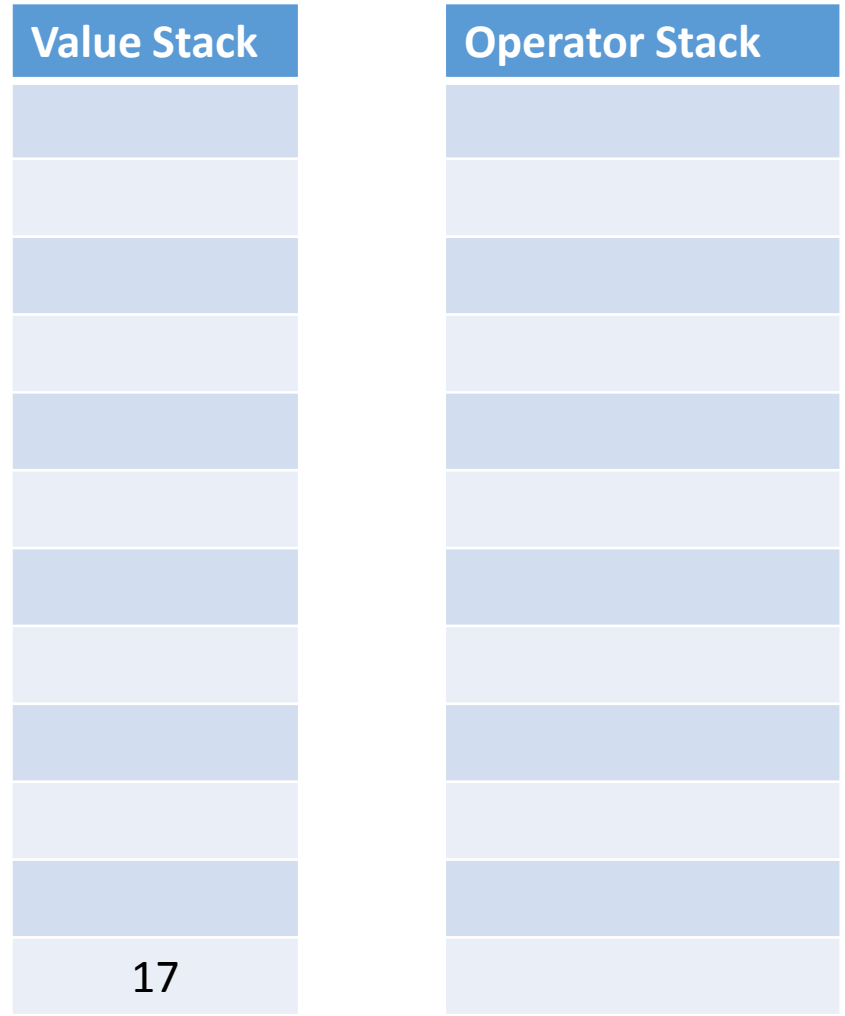
Value Stack	Operator Stack
5	
12	+

Part 2 – Evaluation Algorithm/Example

PopAndEval ( )

Result = 12 + 5 = 17

- Step 1: Op = Top of OP Stack
- Step 2: Pop OP Stack
- Step 3: Val2 = Top of Value Stack
- Step 4: Pop Value Stack
- Step 5: Val1 = Top of Value Stack
- Step 6: Pop Value Stack
- Step 7: Evaluate Result
- Step 8: Push Result on Value Stack



Part 2 – Evaluation Algorithm/Example

Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
^

- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm

- Current Token: EOLN
  - When OP Stack is Empty
  - Expression Result on Top of Value Stack

Value Stack	Operator Stack
17	