

CS 211

Project 5 – Infix Expression Evaluation

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? (perhaps not)
 - one Stack Class of the Token class from the proj5Base.cpp file will work
 - one Stack Class of integers will also work
 - since characters are a sub-type of integers (conversion with type cast may be needed)
 - Use C++ templating is not allowed!!
 - First: An INSANE idea for your first C++ Object program
 - Second: You are to write this code yourself not copy it from the internet

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 1 – Create your own Stack Class

- Naming your class “MyStack” rather than “Stack” is suggested
 - avoids confusion with any system provided stack class
- Dynamic array
 - initialize to only hold 2 values
 - grow by an increase of 2 values each time
 - just like Project 2
- Data Members MUST be private
- In-Class Use Only Methods should also be private
 - ex: grow() method called by push()

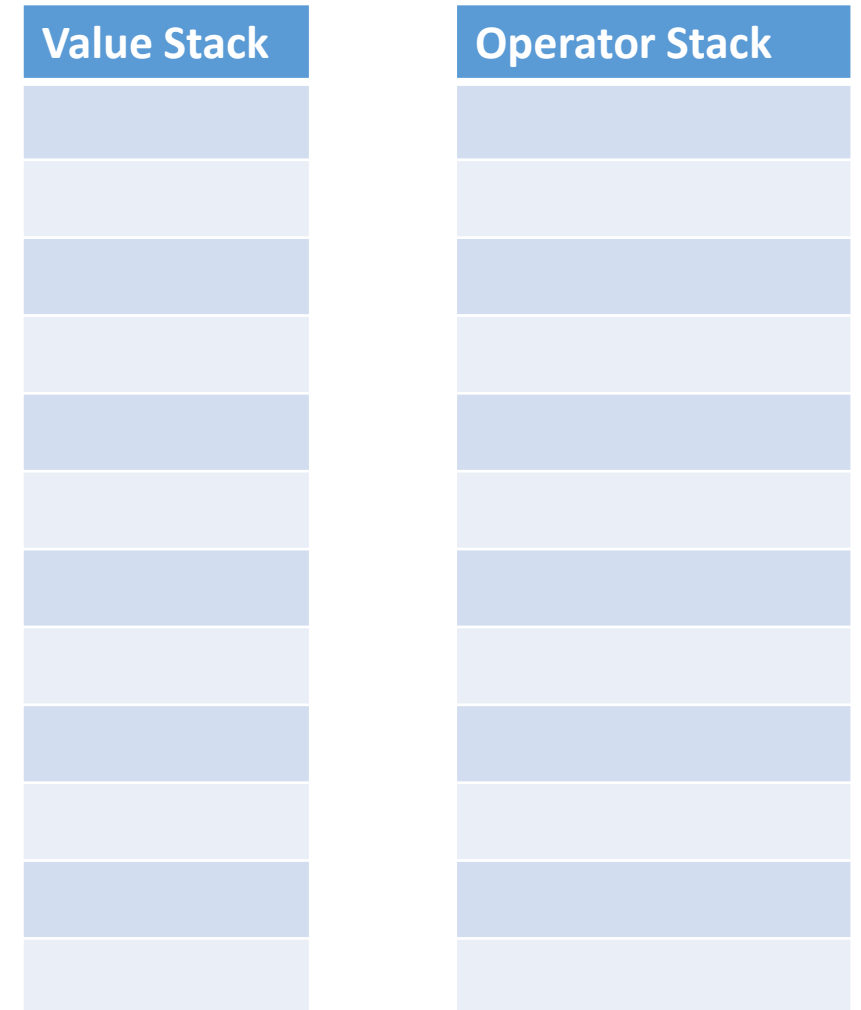
Part 2 – Evaluation Algorithm/Example

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:



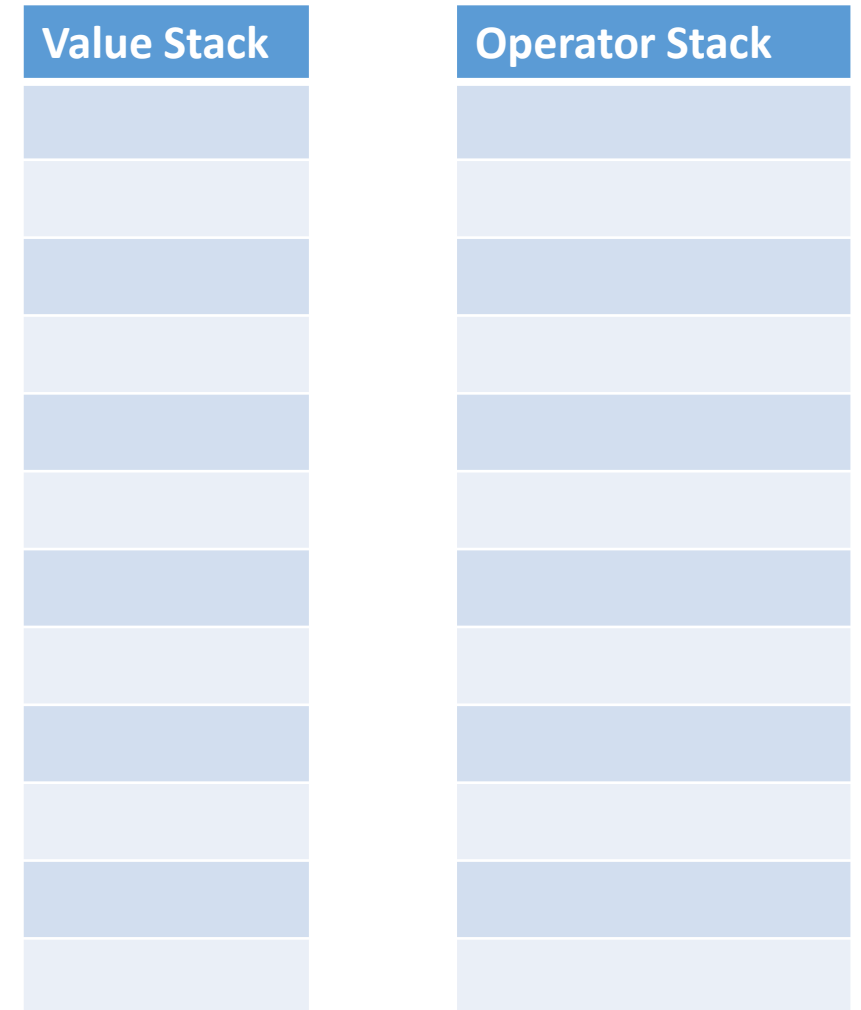
Part 2 – Evaluation Algorithm/Example

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



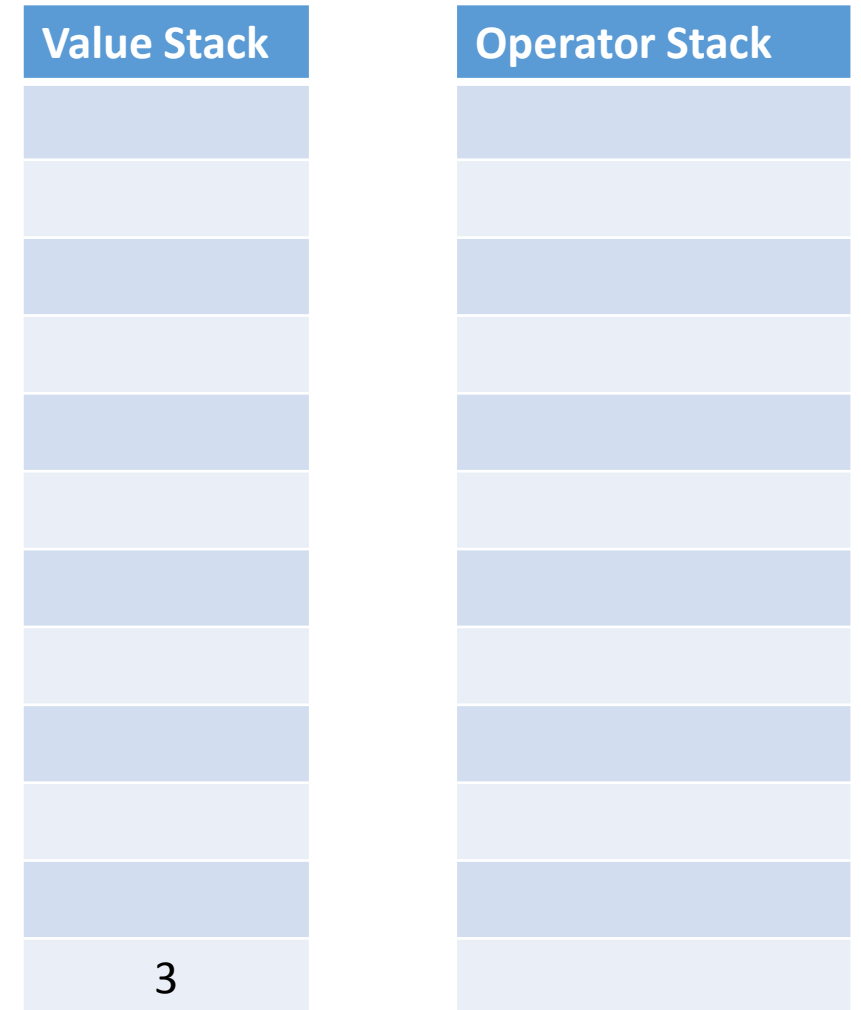
Part 2 – Evaluation Algorithm/Example

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



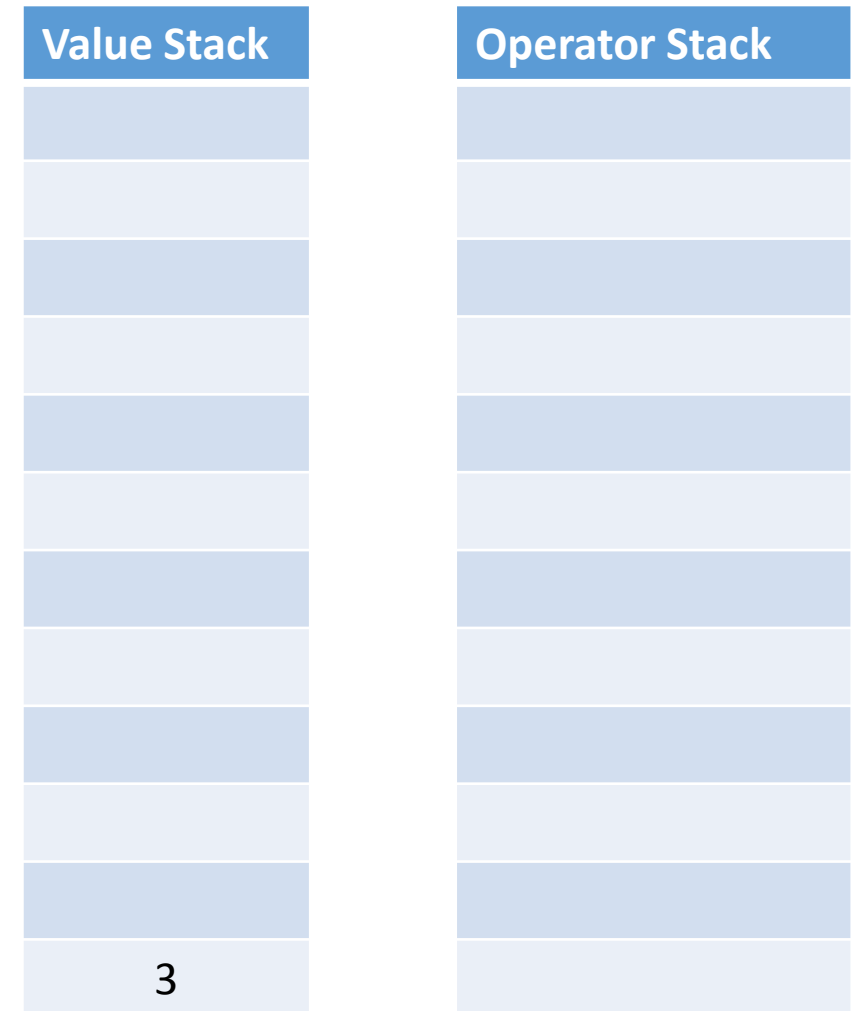
Part 2 – Evaluation Algorithm/Example

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 *



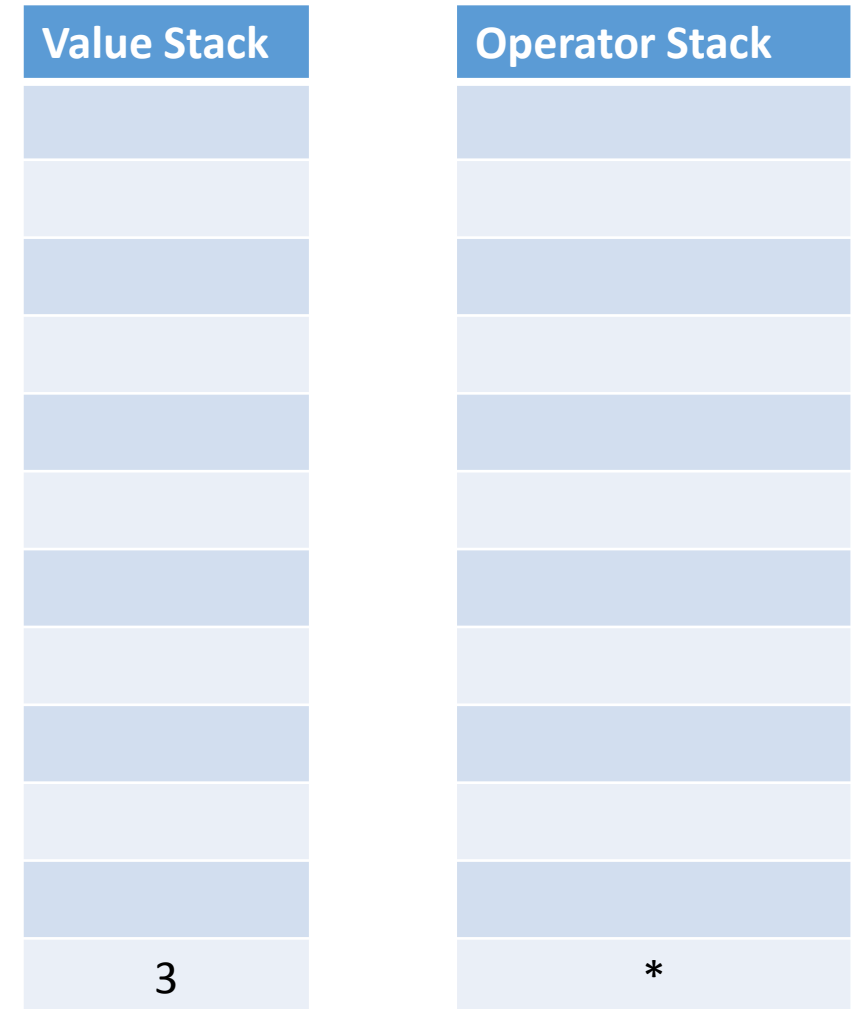
Part 2 – Evaluation Algorithm/Example

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



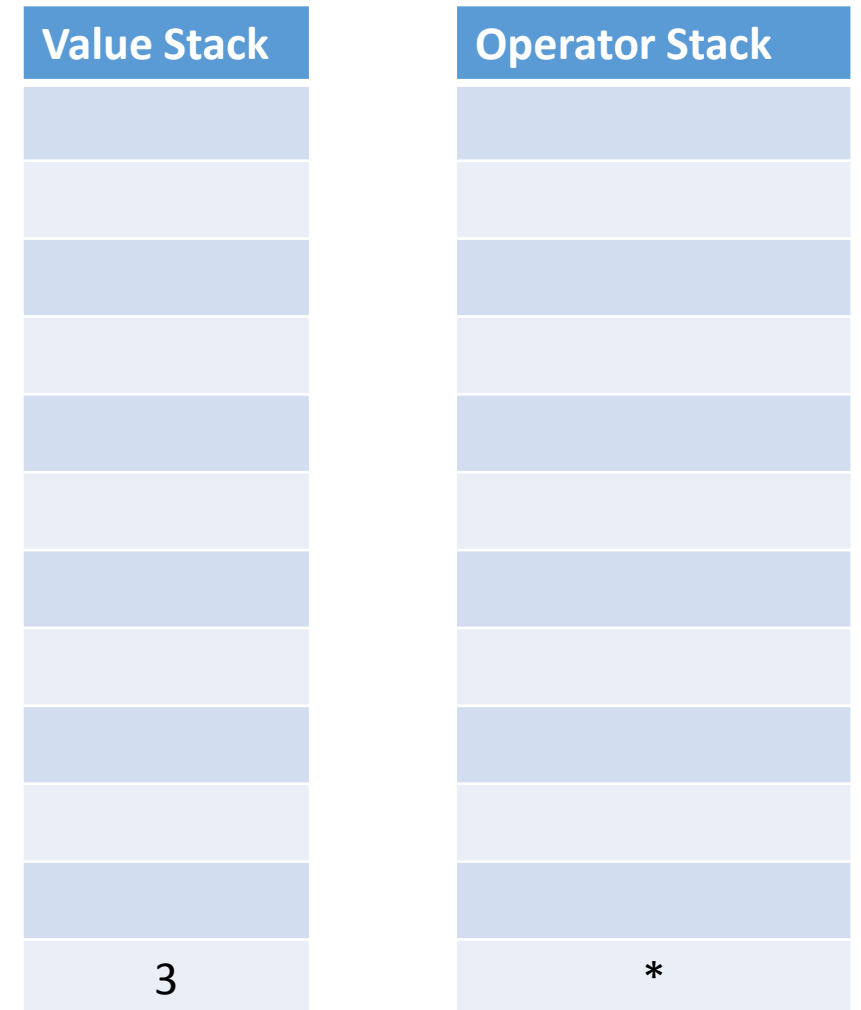
Part 2 – Evaluation Algorithm/Example

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



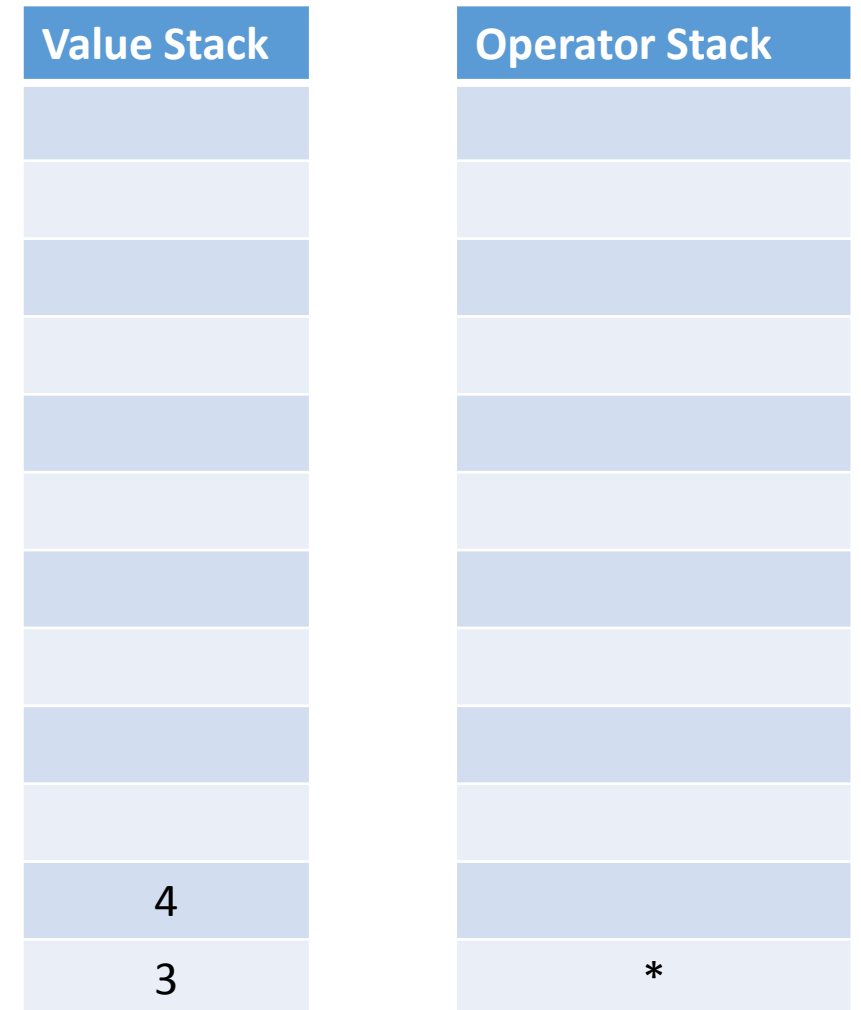
Part 2 – Evaluation Algorithm/Example

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



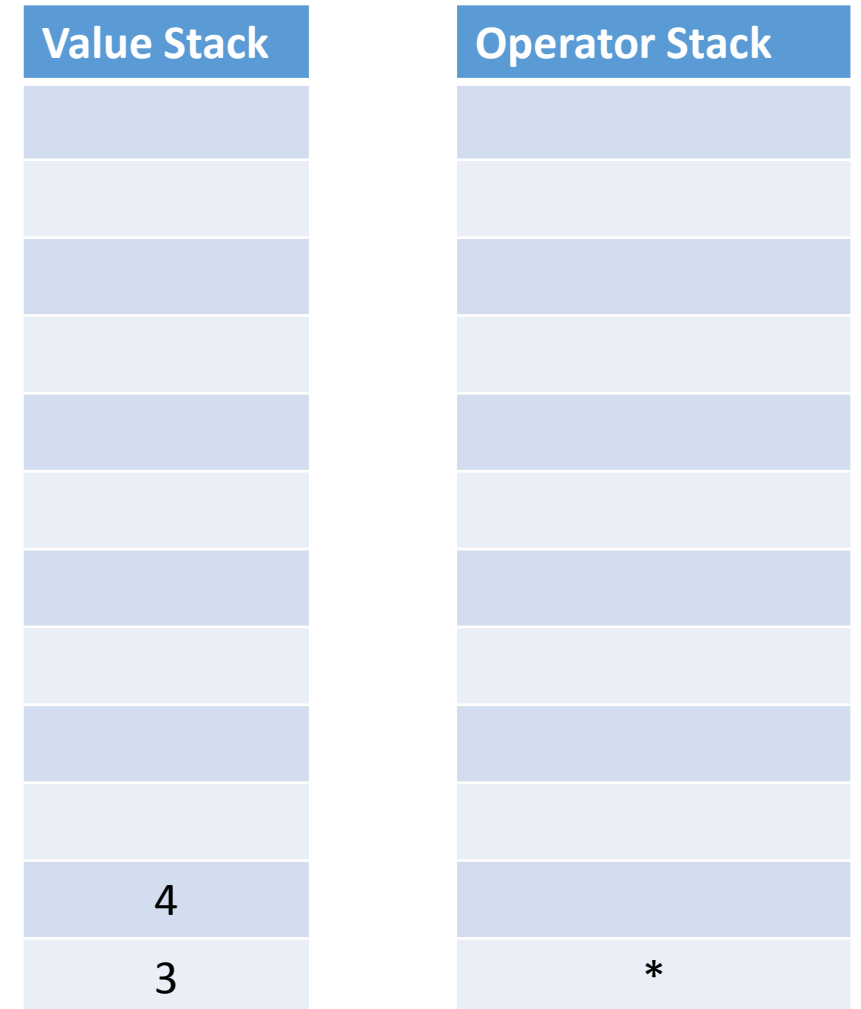
Part 2 – Evaluation Algorithm/Example

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 +



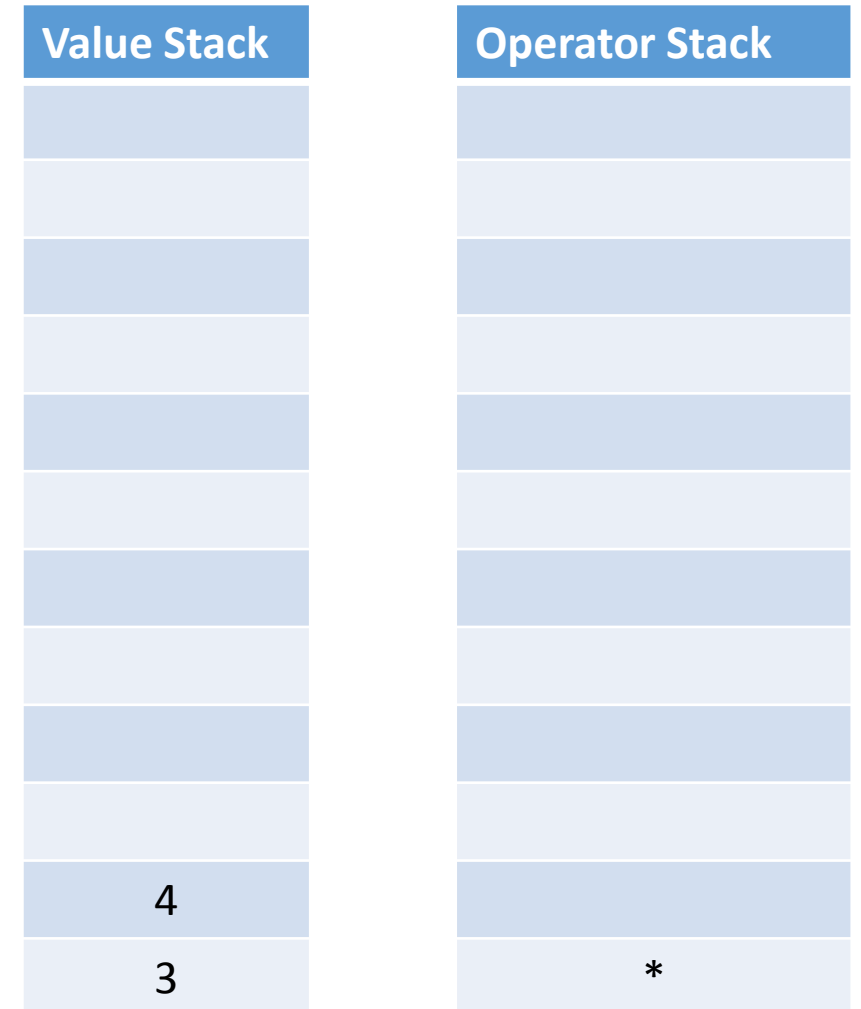
Part 2 – Evaluation Algorithm/Example

Step 1. Create/Clear Value Stack and Operator Stack

Step 2. Get Tokens until End Of Line is Reached

3 * 4 + 5 EOLN
 ^

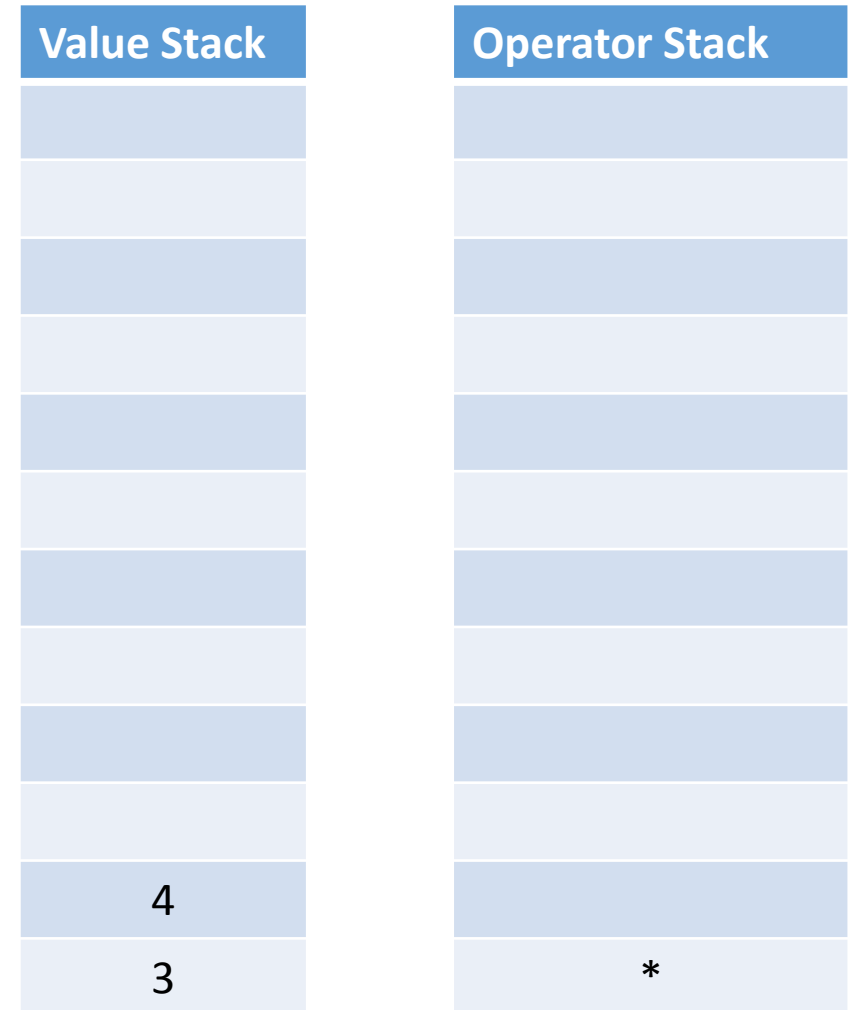
- If token is Value: Push on Value Stack
- If token is Operator: Follow Algorithm
- Current Token: OPERATOR +
 - Since Top OP has higher precedence
PopAndEval() then ...



Part 2 – Evaluation Algorithm/Example

PopAndEval ()

Result = Val1 Op Val2

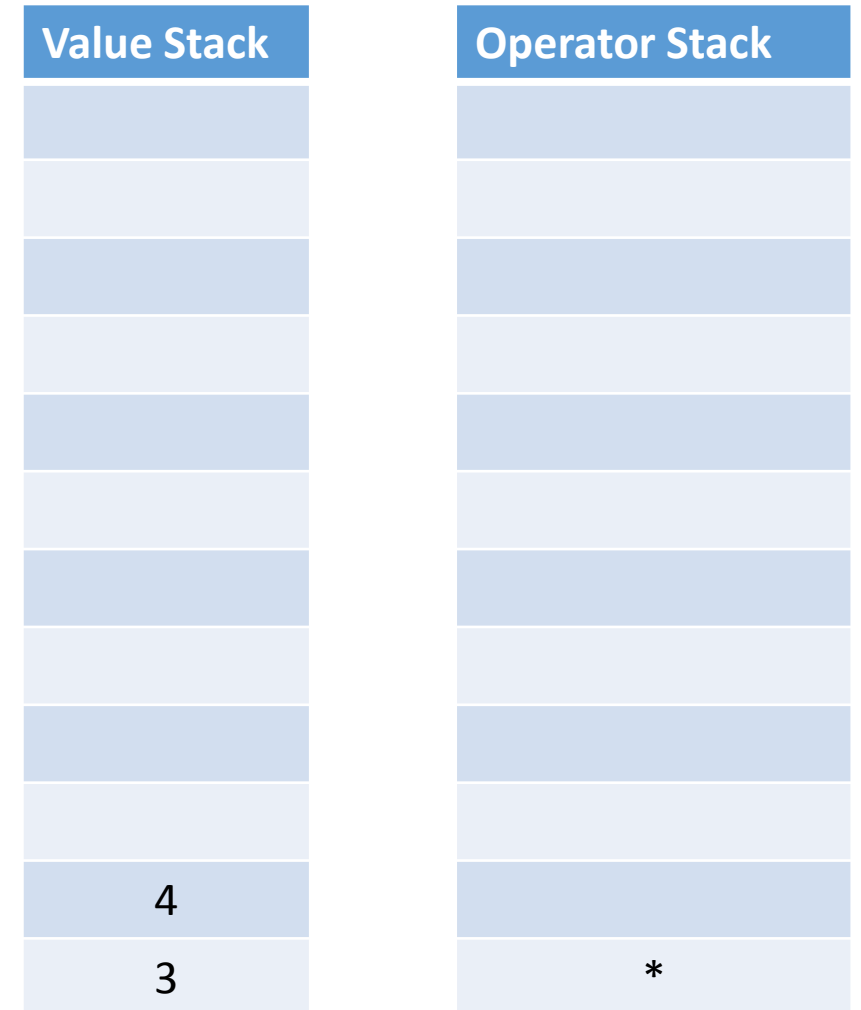


Part 2 – Evaluation Algorithm/Example

PopAndEval ()

Result = Val1 * Val2

- Step 1: Op = Top of OP Stack

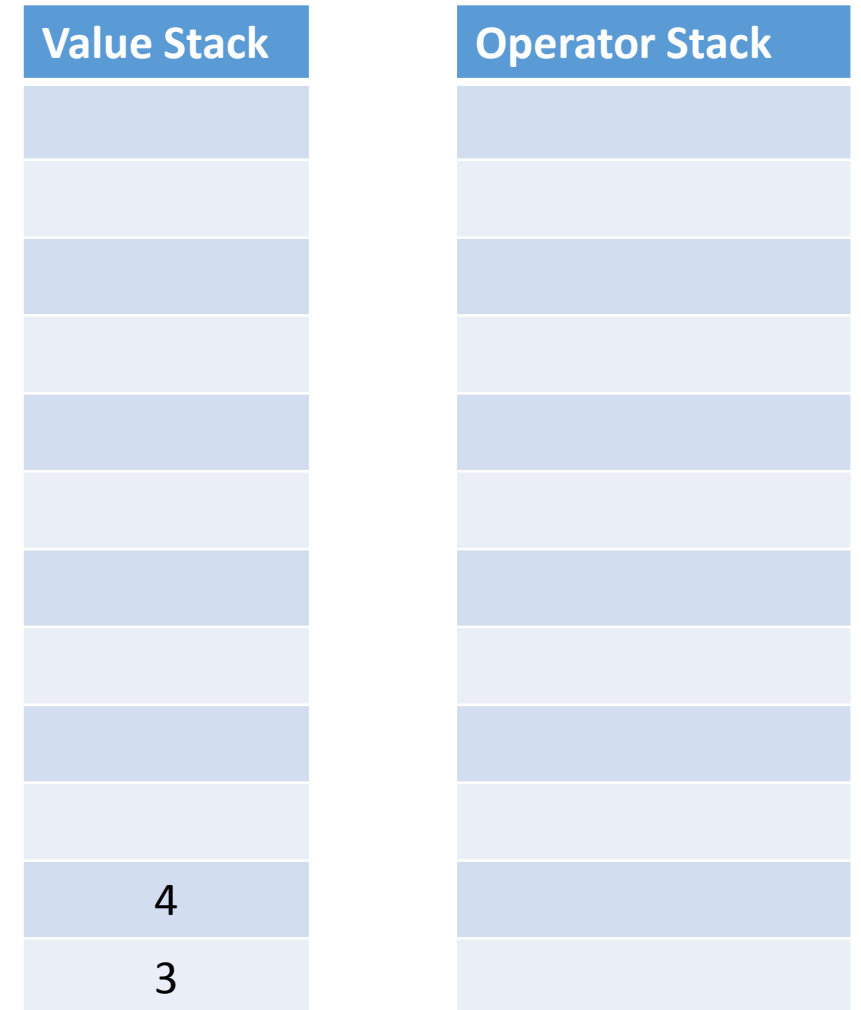


Part 2 – Evaluation Algorithm/Example

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

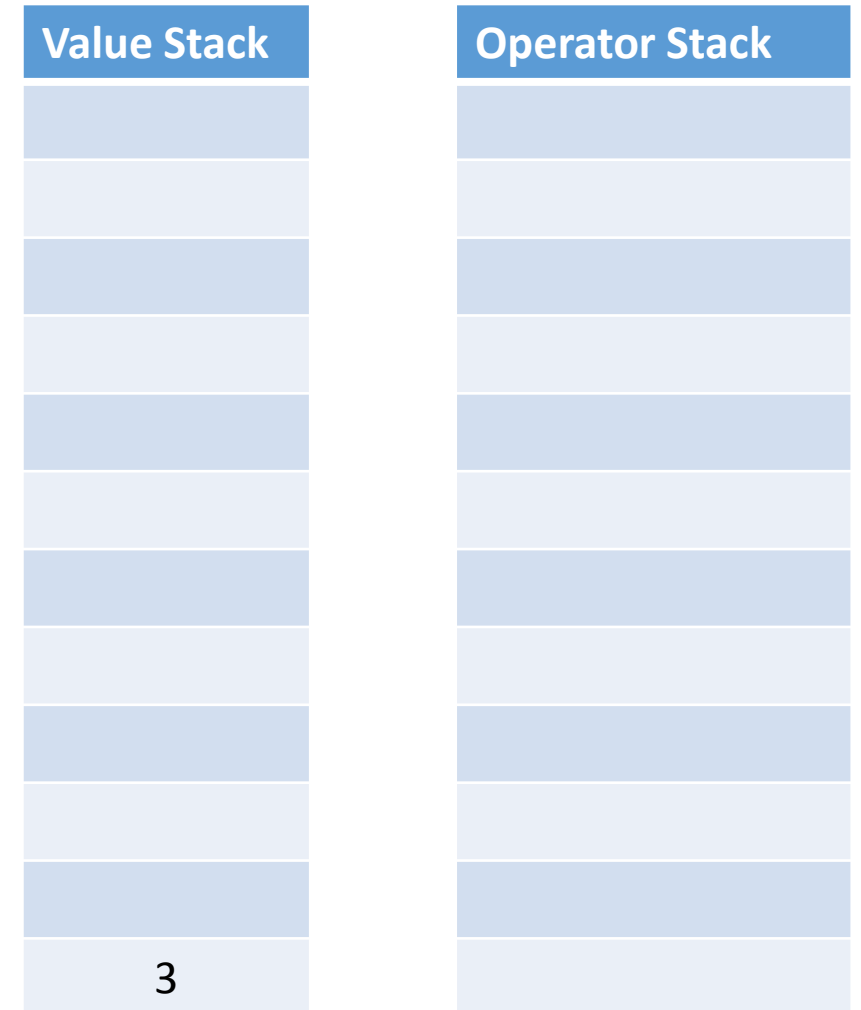


Part 2 – Evaluation Algorithm/Example

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

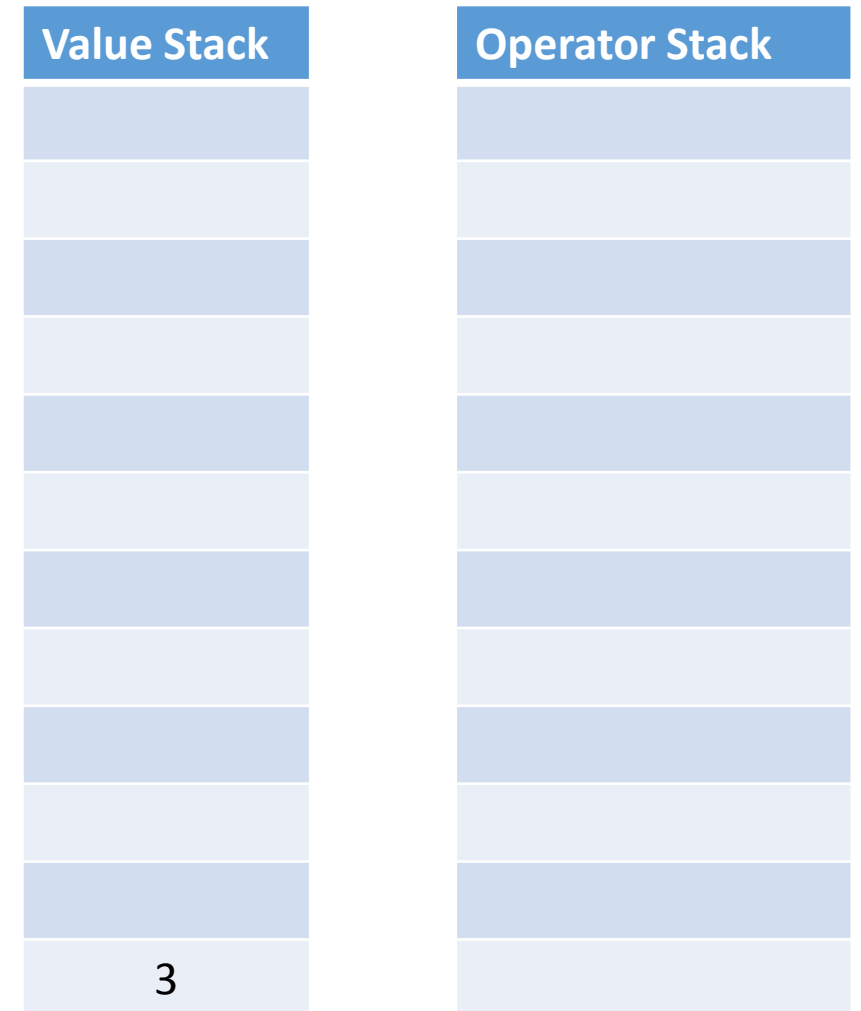


Part 2 – Evaluation Algorithm/Example

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

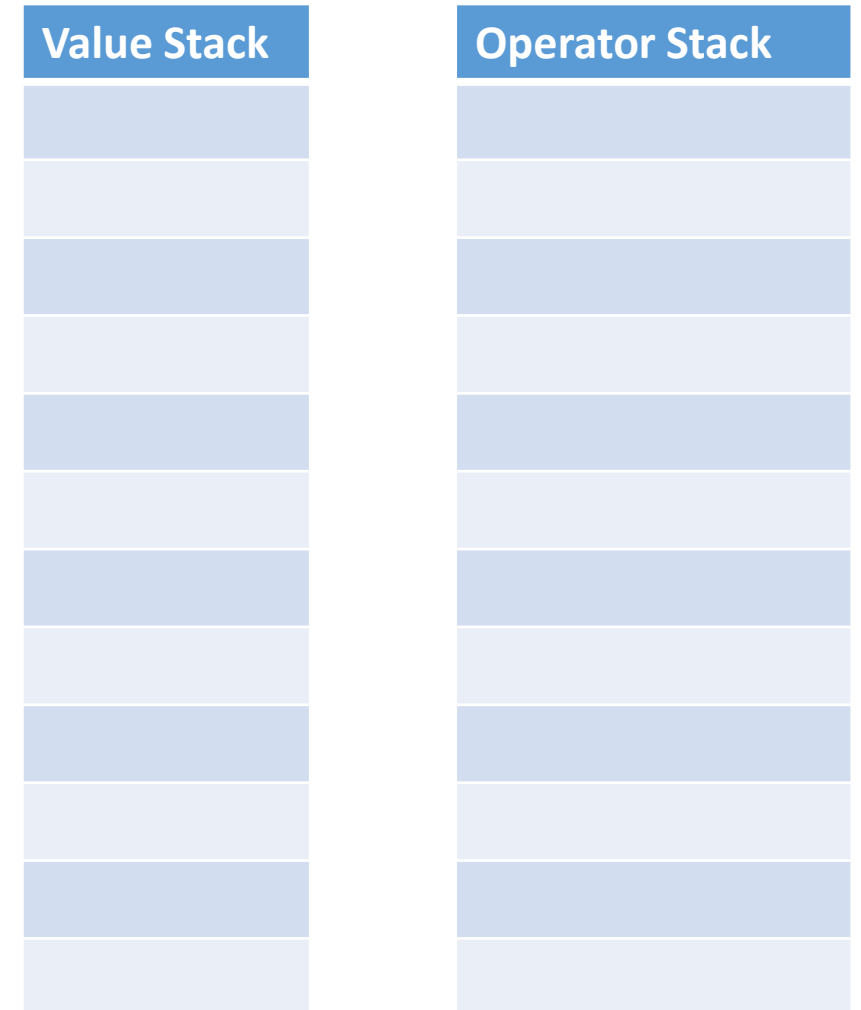


Part 2 – Evaluation Algorithm/Example

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

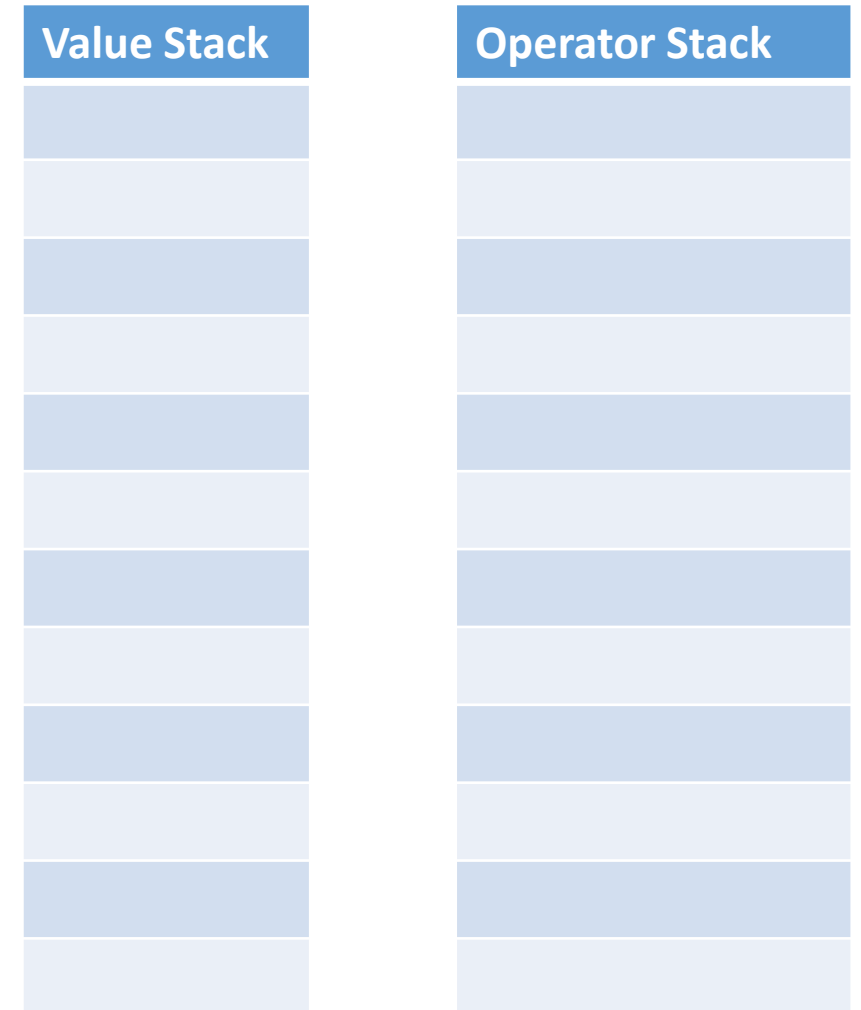


Part 2 – Evaluation Algorithm/Example

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

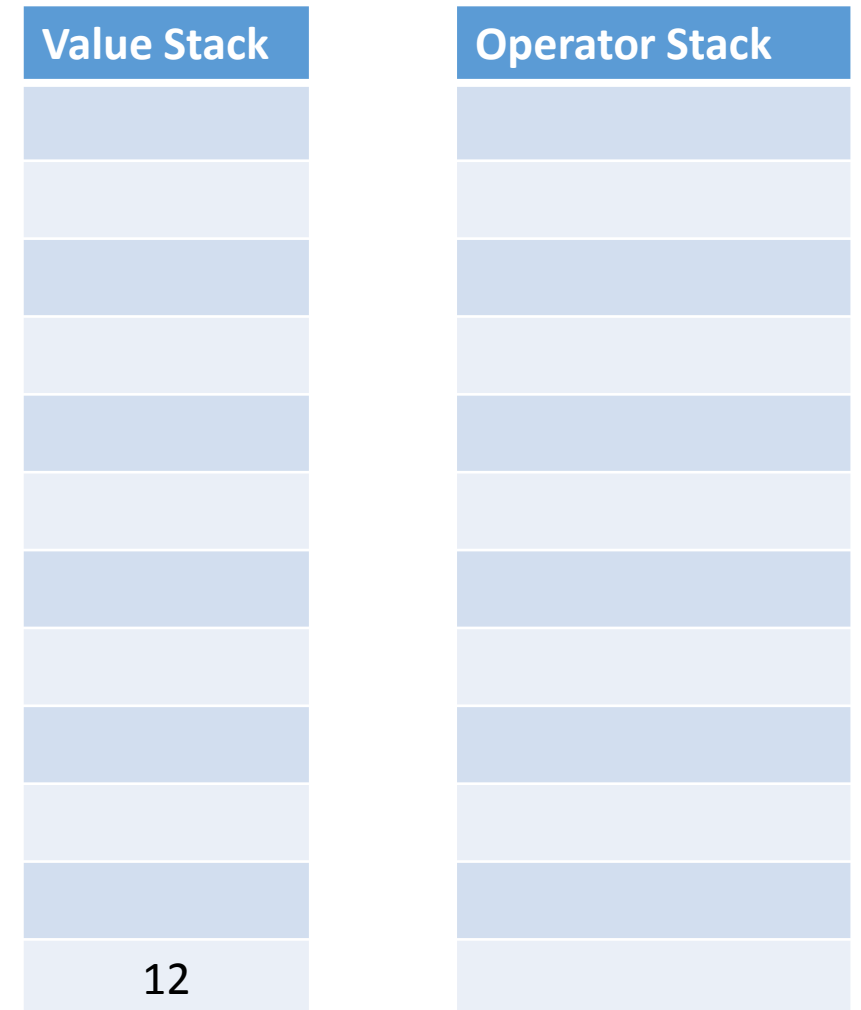


Part 2 – Evaluation Algorithm/Example

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



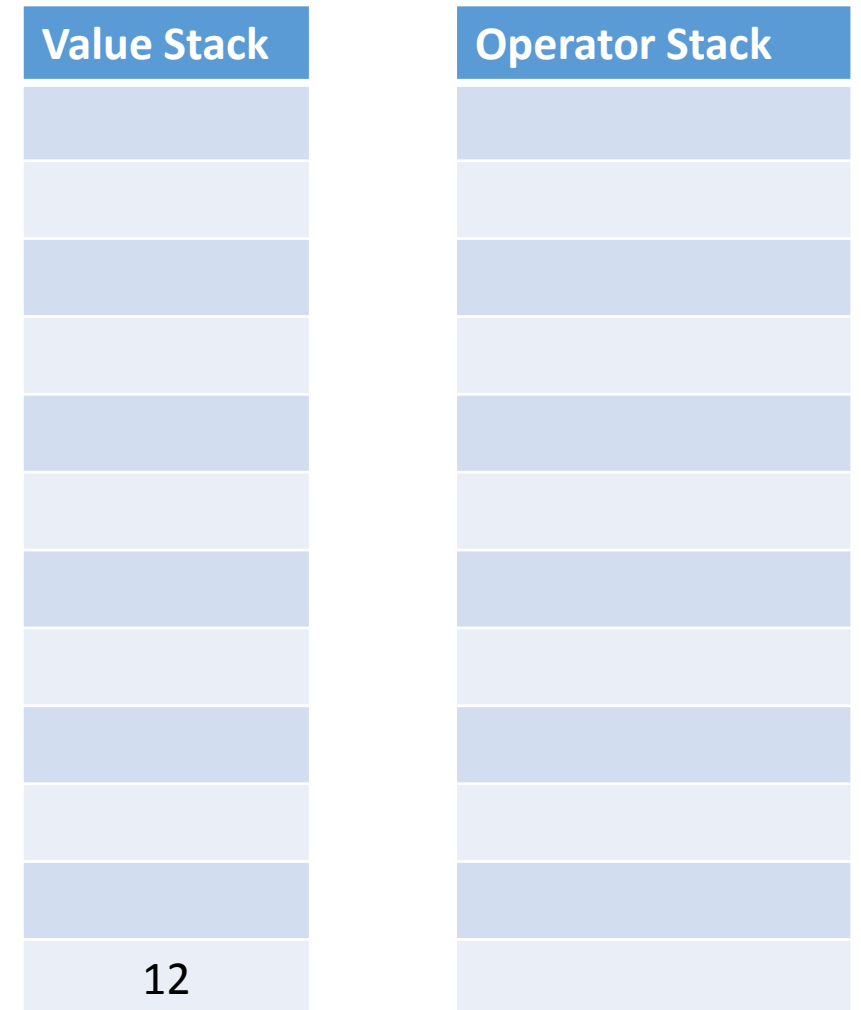
Part 2 – Evaluation Algorithm/Example

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 ...



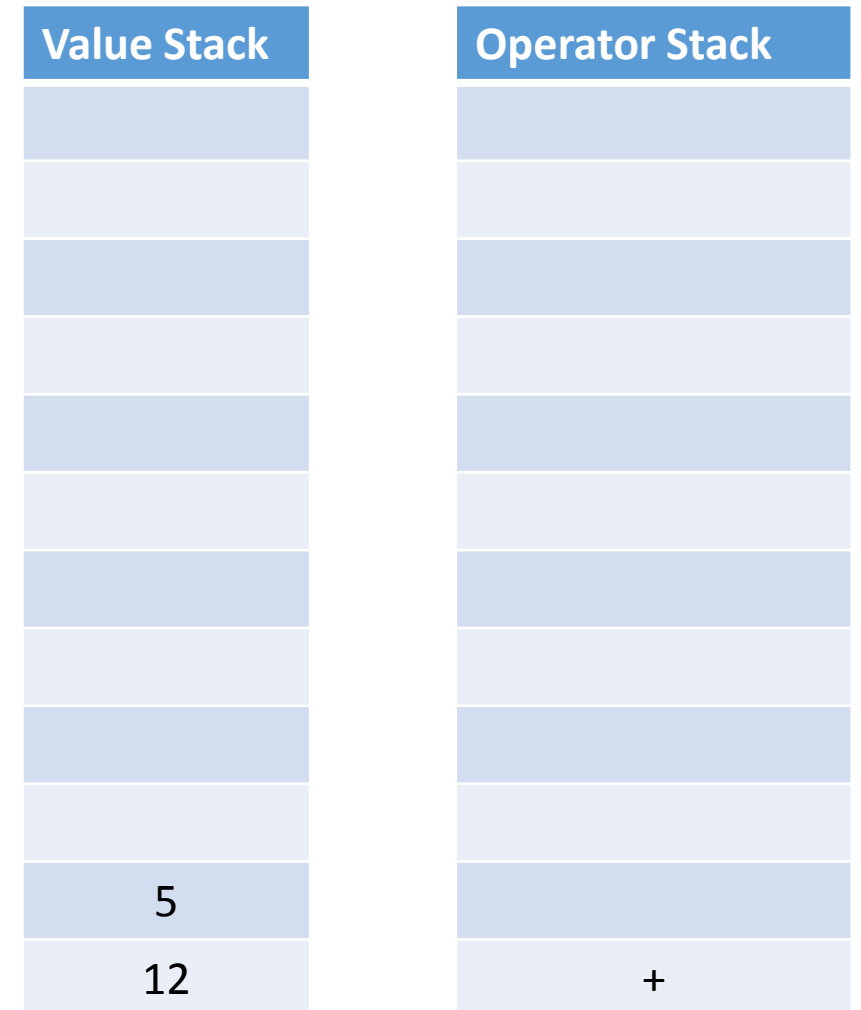
Part 2 – Evaluation Algorithm/Example

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



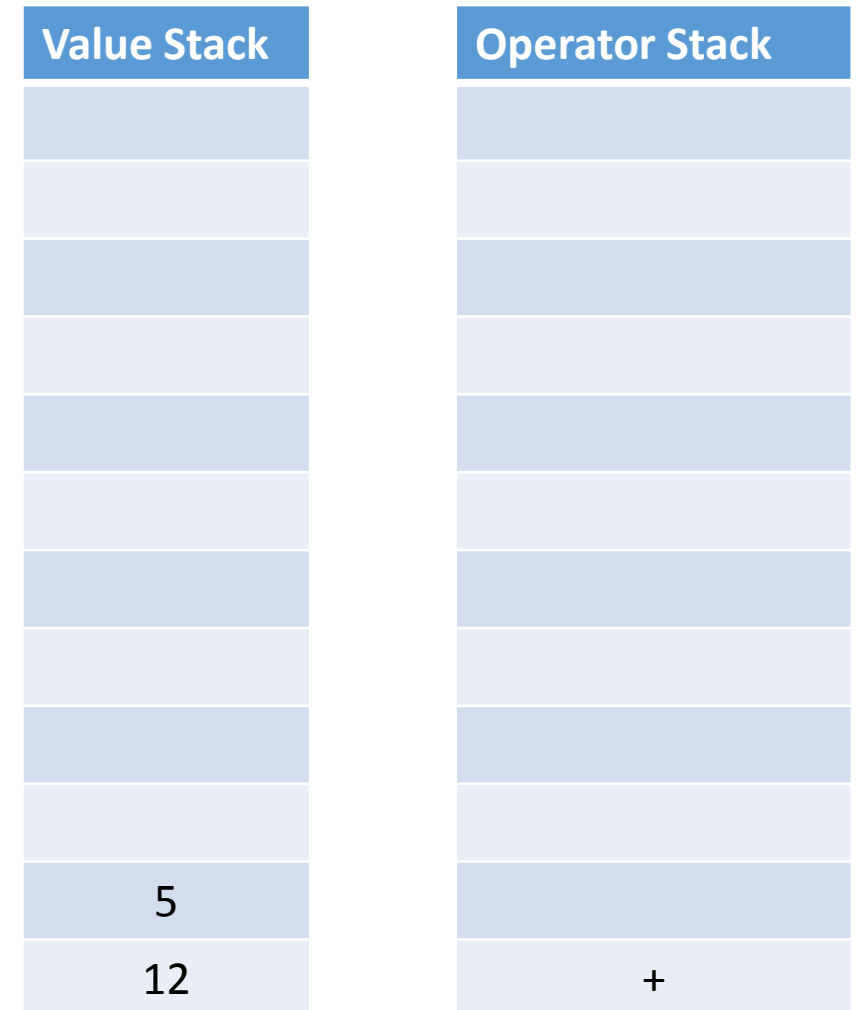
Part 2 – Evaluation Algorithm/Example

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

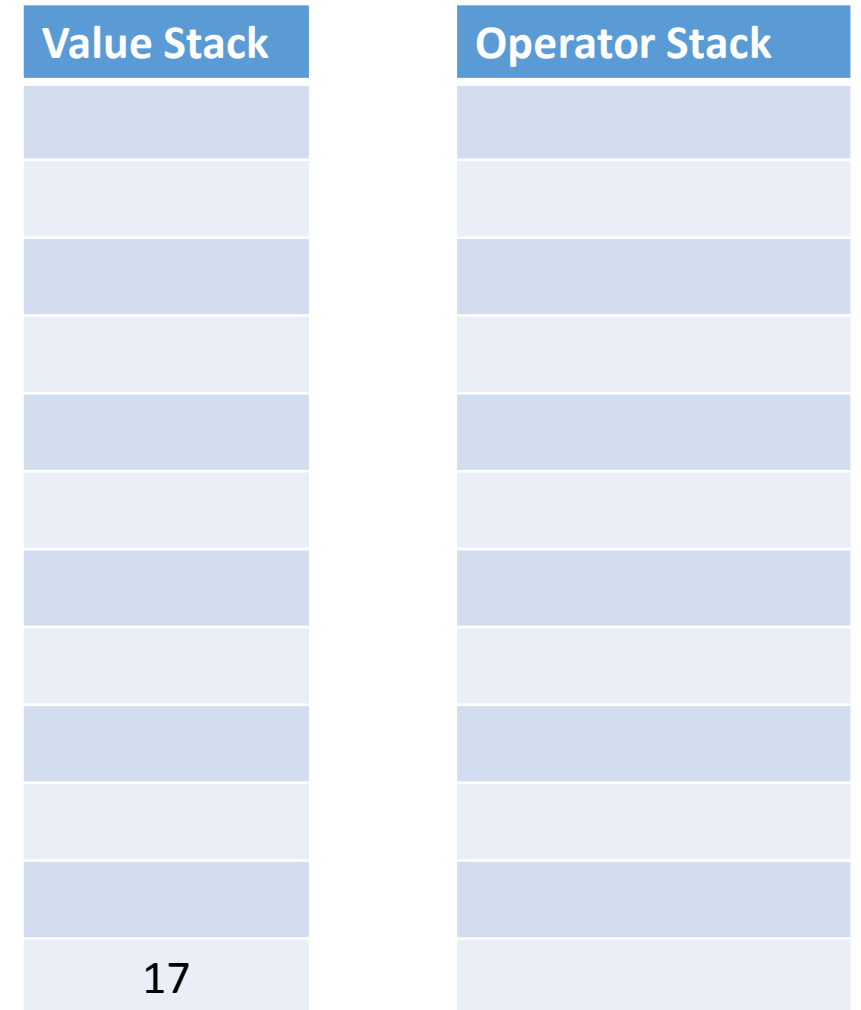


Part 2 – Evaluation Algorithm/Example

PopAndEval ()

$$\text{Result} = 12 + 5 = 17$$

- 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

