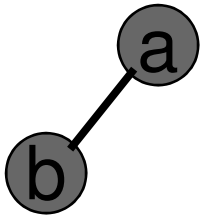
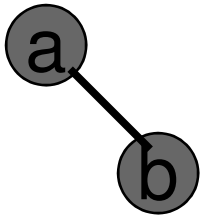
Binary Tree Construction

- Suppose that the elements in a binary tree are distinct.
- Can you construct the binary tree from which a given traversal sequence came ?
  - When a traversal sequence has more than one element, the binary tree is not uniquely defined.
  - Therefore, the tree from which the sequence was obtained cannot be reconstructed uniquely.

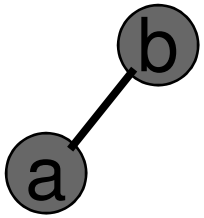
Examples: binary tree is not uniquely defined with a given traversal

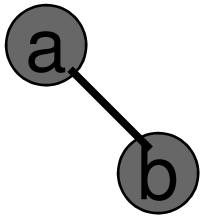
preorder = ab



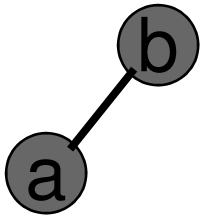


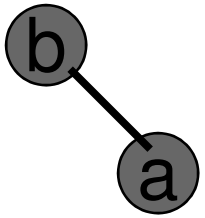
inorder = ab



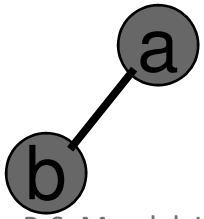


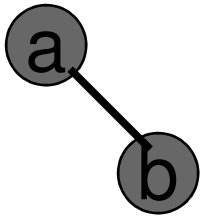
postorder = ab





level order = ab





Binary Tree Construction

- Can you construct the binary tree, given two traversal sequences?
  - Depends on which two sequences are given.

Preorder and Postorder

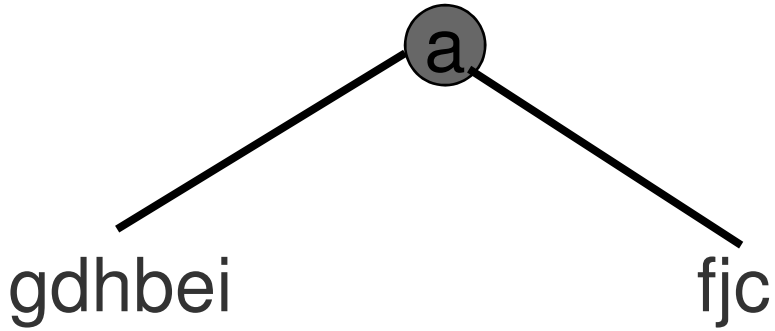
preorder = ab

postorder = ba

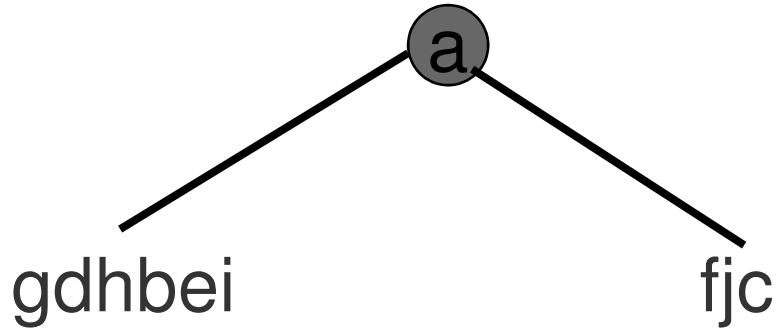
- Preorder and postorder do not uniquely define a binary tree.
- Nor do preorder and level order (same example).
- Nor do postorder and level order (same example).

Inorder and Preorder

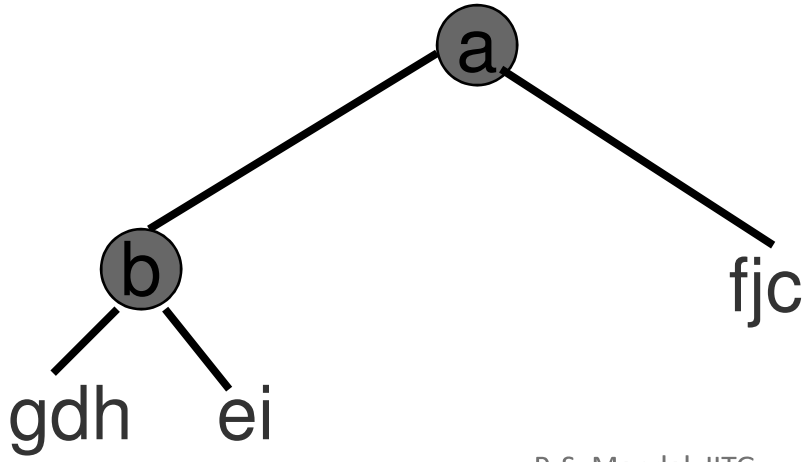
- inorder = g d h b e i a f j c
- preorder = a b d g h e i c f j
- Scan preorder from left to right using the inorder to separate left and right subtrees.
- a is the root of the tree; gdhbei are in the left subtree; fjc are in the right subtree.



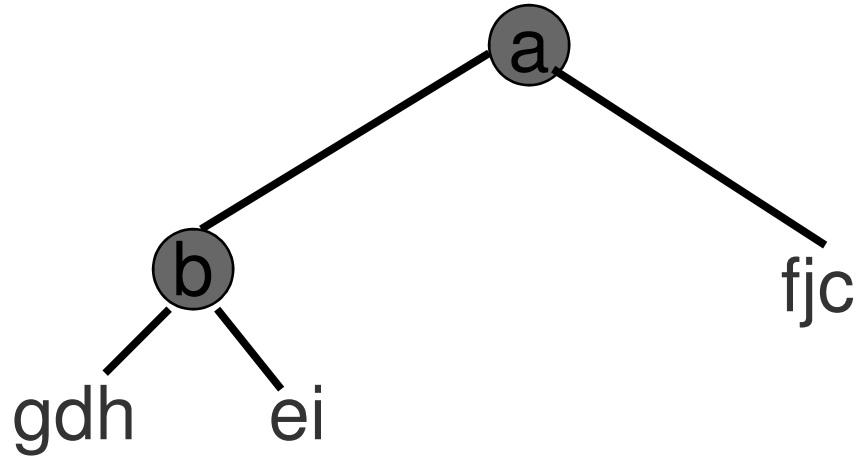
Inorder and Preorder



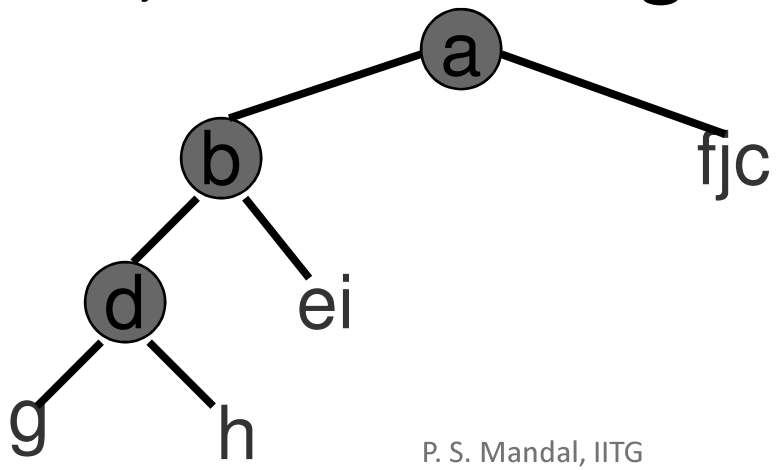
- preorder = a b d g h e i c f j
- b is the next root; gdh are in the left subtree; ei are in the right subtree.



Inorder and Preorder



- preorder = a b d g h e i c f j
- d is the next root; g is in the left subtree; h is in the right subtree.



Inorder and Postorder

- Scan postorder from right to left using inorder to separate left and right subtrees.
- inorder = g d h b e i a f j c
- postorder = g h d i e b j f c a
- Tree root is a; gdhbei are in left subtree; fjc are in right subtree.

Inorder and Level Order

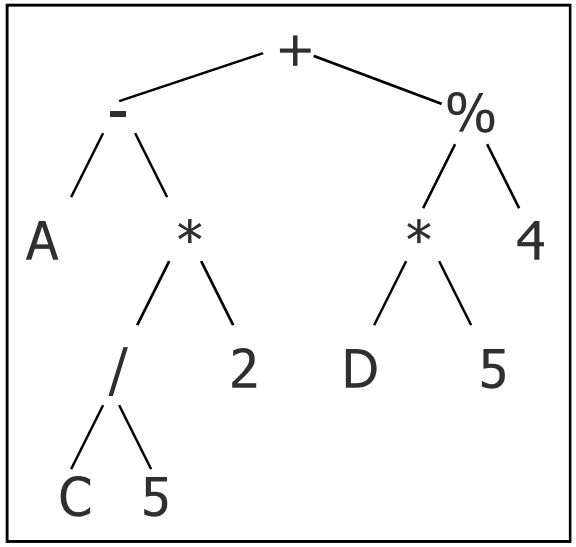
- Scan level order from left to right using inorder to separate left and right subtrees.
- inorder = g d h b e i a f j c
- level order = a b c d e f g h i j
- Tree root is a; gdhbei are in left subtree; fjc are in right subtree.

Uniquely define a binary tree

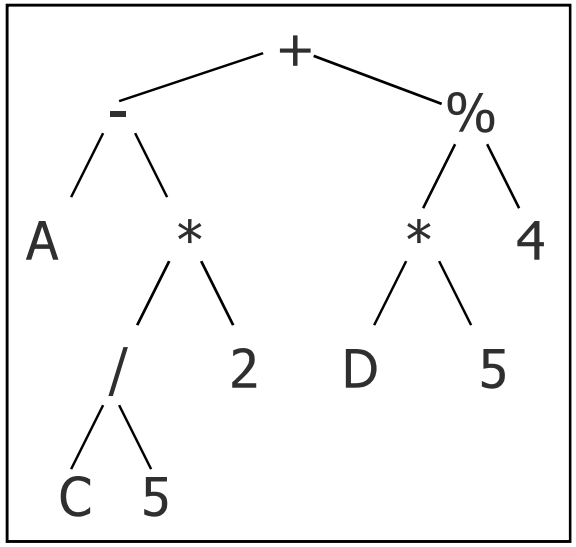
- for given two traversal sequences
  - Preorder and postorder X
  - Preorder and level order X
  - Postorder and level order X
  - Indoder and preorder √
  - Inorder and postorder √
  - Inorder and level order √

Expression Trees

- Expressions, programs, etc can be represented by tree structures
- It's a binary tree
- The leaves of the expression tree are operands
- The other nodes are contain operators
  - **E.g. Arithmetic Expression Tree**
  - **A-(C/5*2)+(D*5%4)**



Tree Traversal

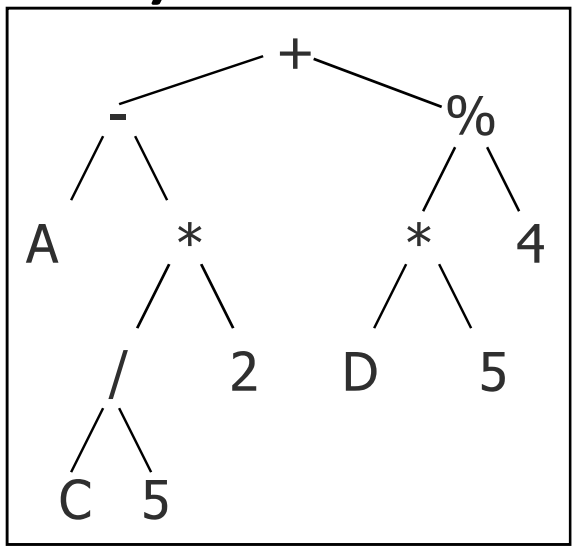


- Goal: visit every node of a tree
- **in-order** traversal

```
void Node::inOrder () {
   if (left != NULL) left->inOrder();
   cout << data << endl;
   if (right != NULL) right->inOrder()
}
```

Output: A – C / 5 * 2 + D * 5 % 4

Tree Traversal (contd.)



- **pre-order** and **post-order**:

```
void Node::preOrder () {
   cout << data << endl;
   if (left != NULL)   left->preOrder ();
   if (right != NULL) right->preOrder ();
}
```

Output: + - A * / C 5 2 % * D 5 4

```
void Node::postOrder () {
   if (left != NULL)   left->preOrder ();
   if (right != NULL) right->preOrder ();
   cout << data << endl;
}
```

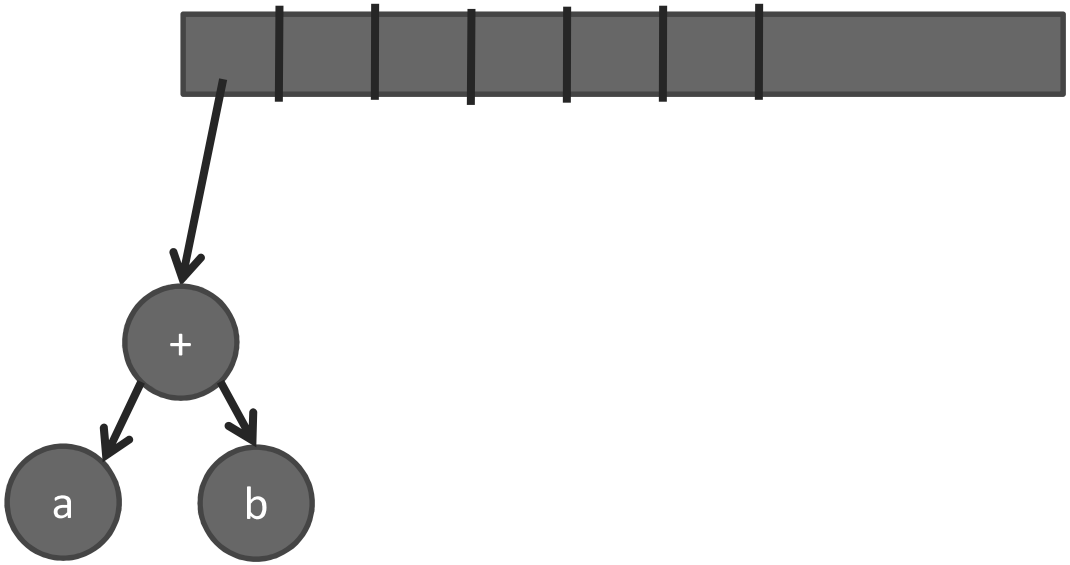
Output: A C 5 / 2 * - D 5 * 4 % +

Constructing an expression Tree

- Convert postfix expression to expression tree
- a b + c d e + * *

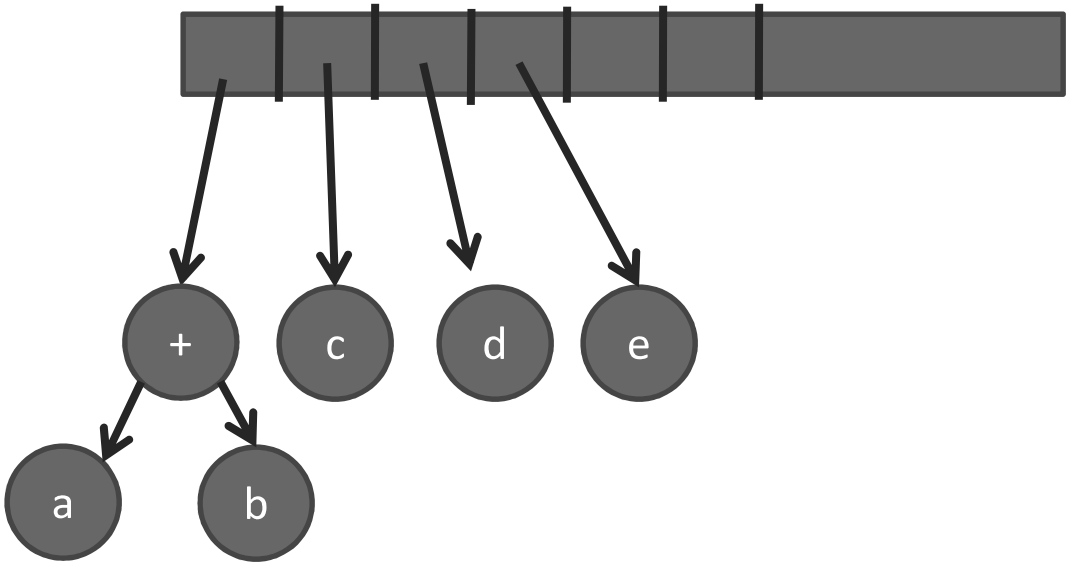
Constructing an expression Tree

- Convert postfix expression to expression tree
- a b + c d e + * *



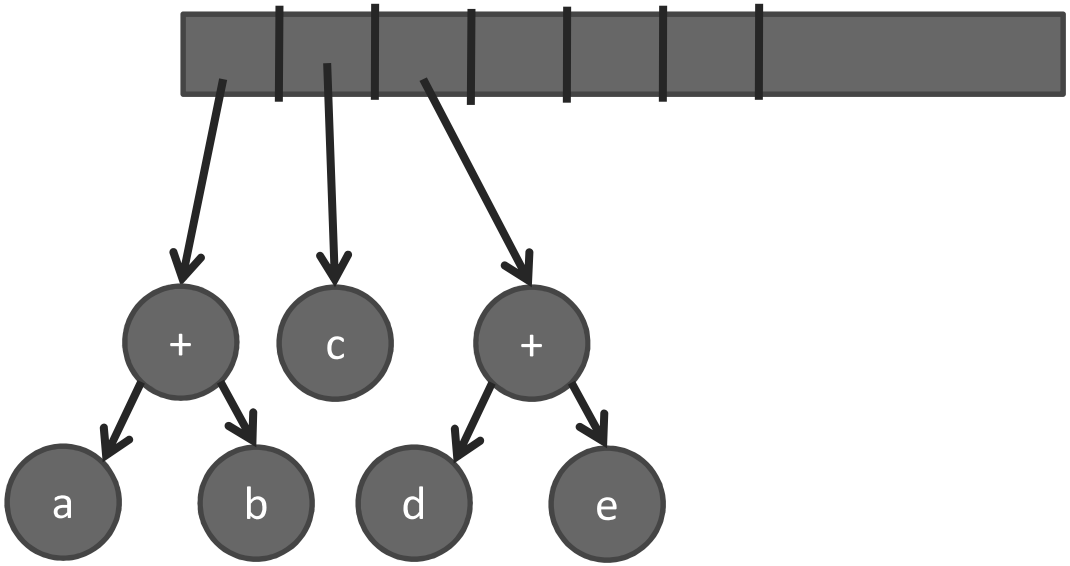
Constructing an expression Tree

- Convert postfix expression to expression tree
- a b + c d e + * *



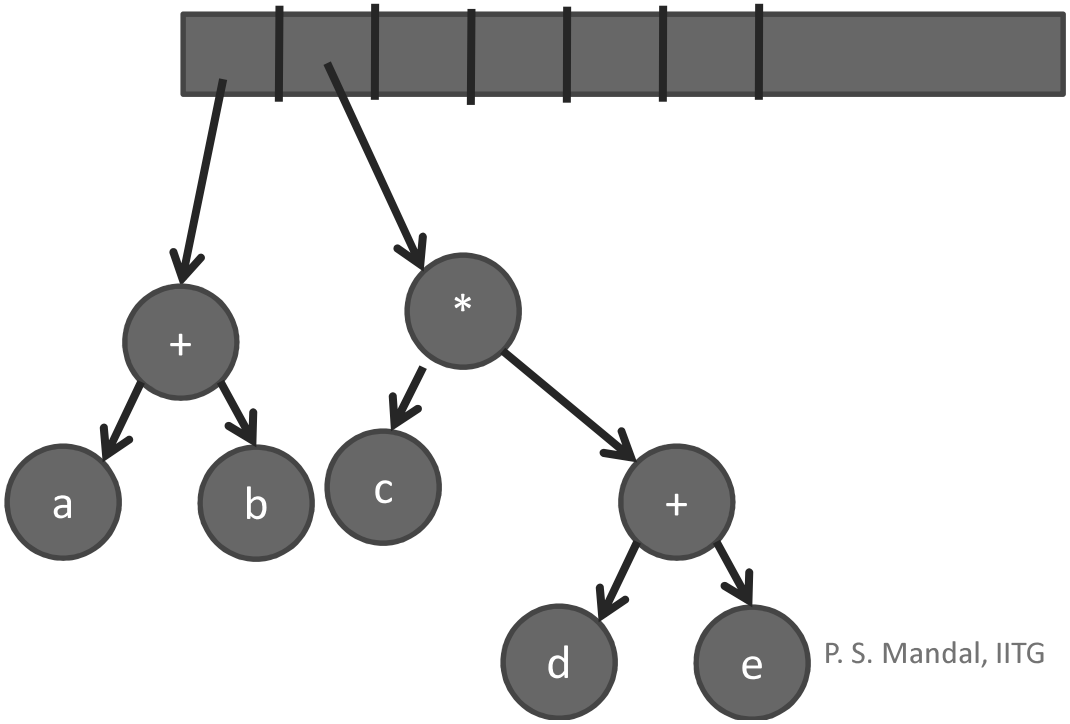
Constructing an expression Tree

- Convert postfix expression to expression tree
- a b + c d e + * *



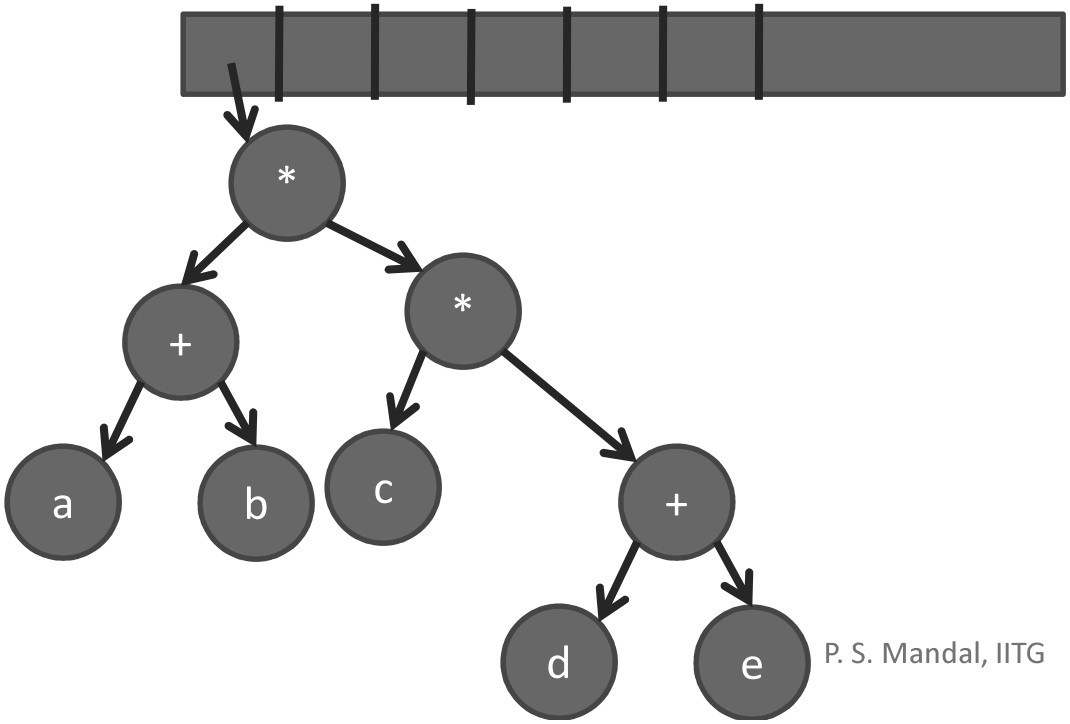
Constructing an expression Tree

- Convert postfix expression to expression tree
- a b + c d e + * *



Constructing an expression Tree

- Convert postfix expression to expression tree
- a b + c d e + * *

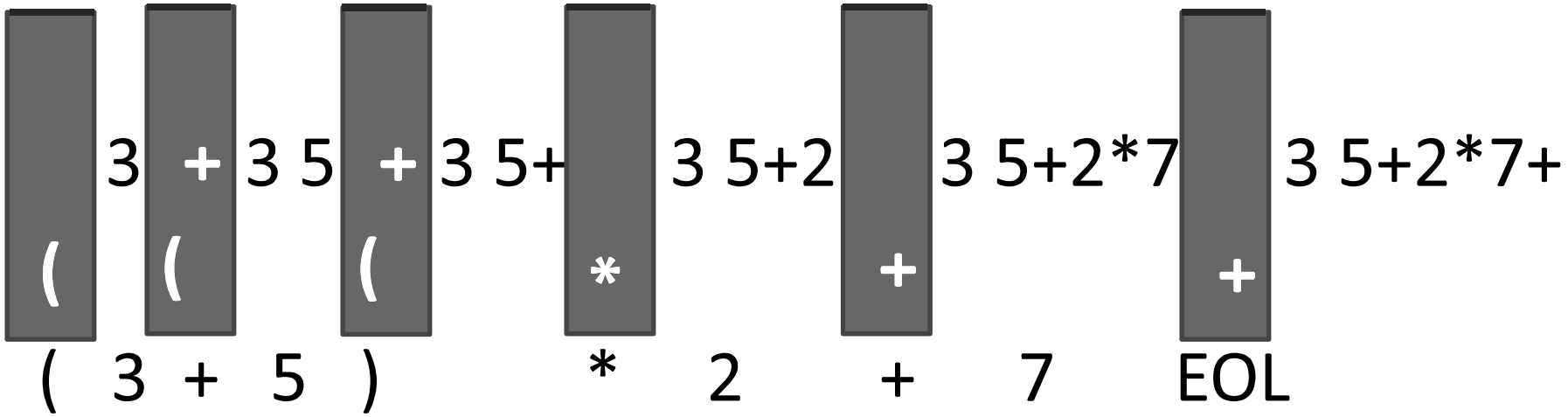


Infix to Posfix

- ***Note*:** input of an expression tree is a postfix expression.
- How we can convert infix expression to postfix expression ?
  - We can use stacks to do so

Infix to Posfix Example

- Infix expression: (3 + 5) * 2 + 7
- The operator stack holds just the operators. Operands are sent to the output directly.



- Postfix expression: 3 5 + 2 * 7 +