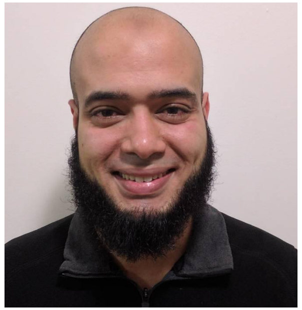
Data Structures
Binary Tree Traversal 2

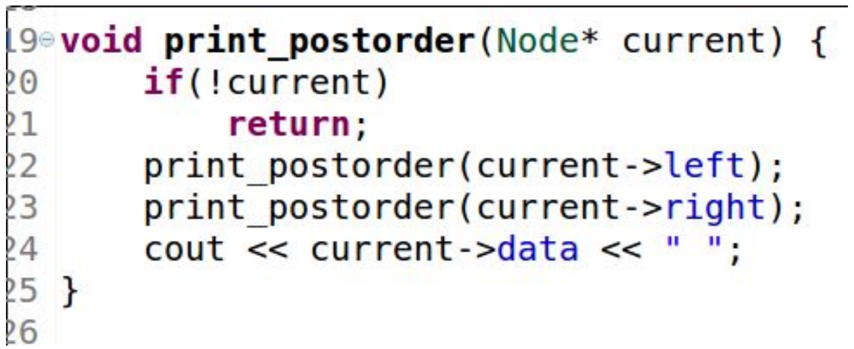
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

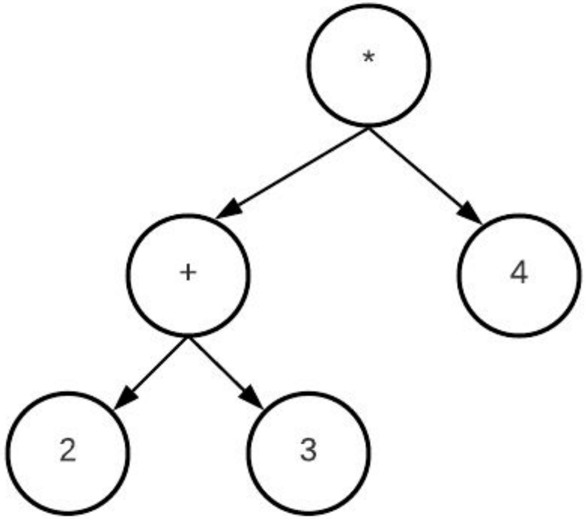


Print Expression Tree: (2 + 3) * 4

- We need to think:
  - Print left subtree
  - Print right subtree
  - Print me

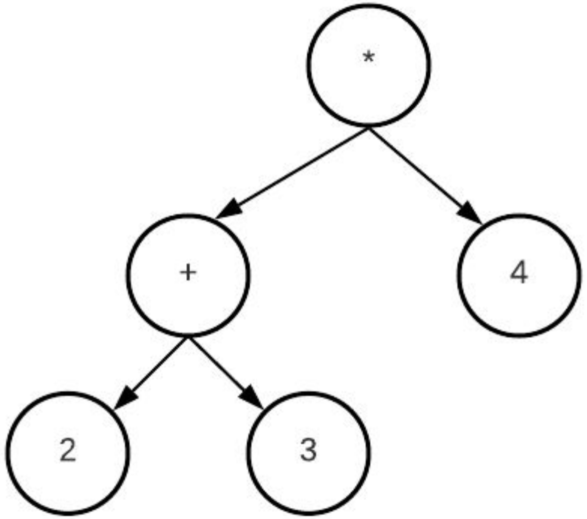


```
void print_postorder(Node* current) {
    if(!current)
        return;
    print_postorder(current->left);
    print_postorder(current->right);
    cout << current->data << " ";
}
```



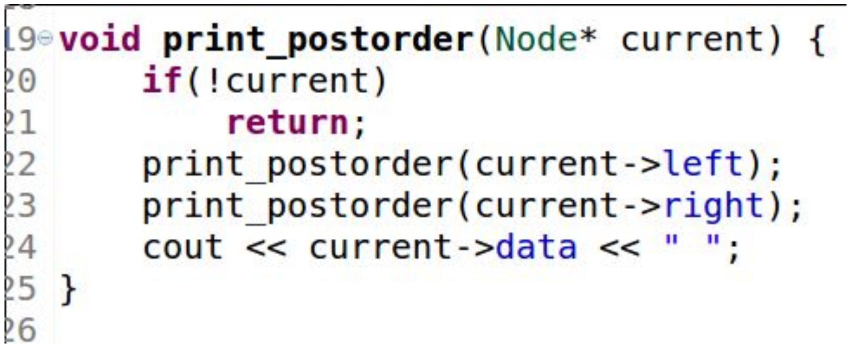
Proper Recursion Tracing

- WHAT not how!
- What is the postfix of (2 + 3) * 4?
  - 2 3 + 4 *
- What is output of print_postorder?
  - Given expression ⇒ prints its post-order
- What is post-order of subtree ‘+’
  - As we did: 2 3 +
- What is post-order of tree ‘*’
  - L = postorder(+) = 2 3 +
  - R = postorder(4) = 4
  - V = *
  - **In total: 2 3 + 4 ***

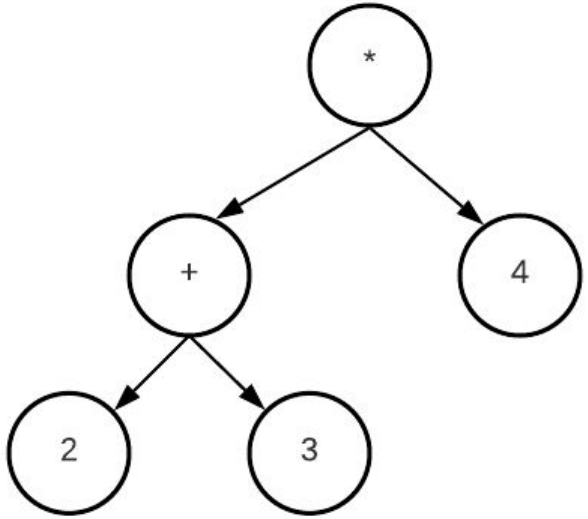


Proper Recursion Tracing

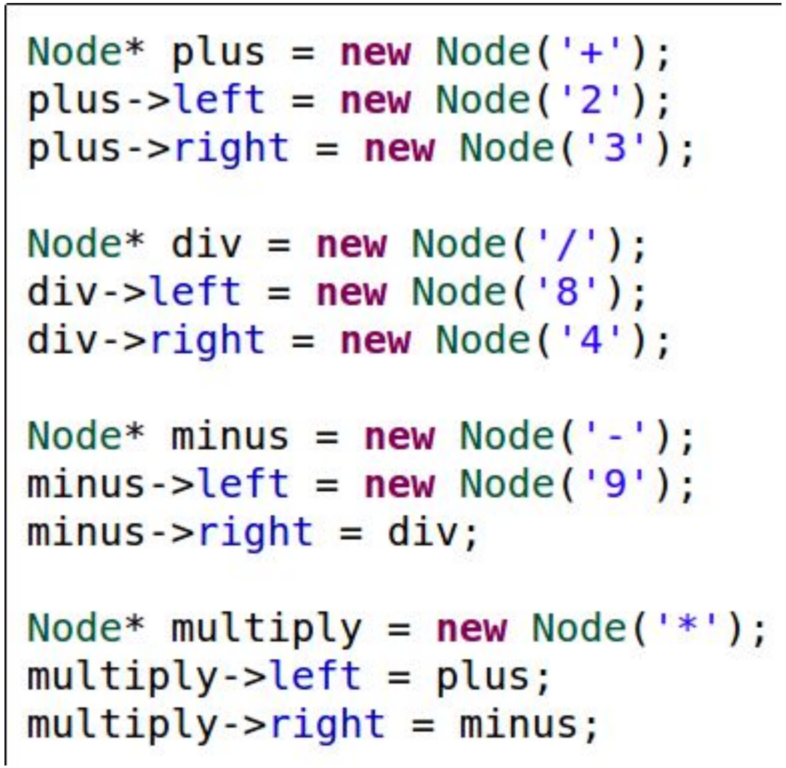
- What:
  - + subtree ⇒ 2 3 +
  - *   tree ⇒ 2 3 + 4 *



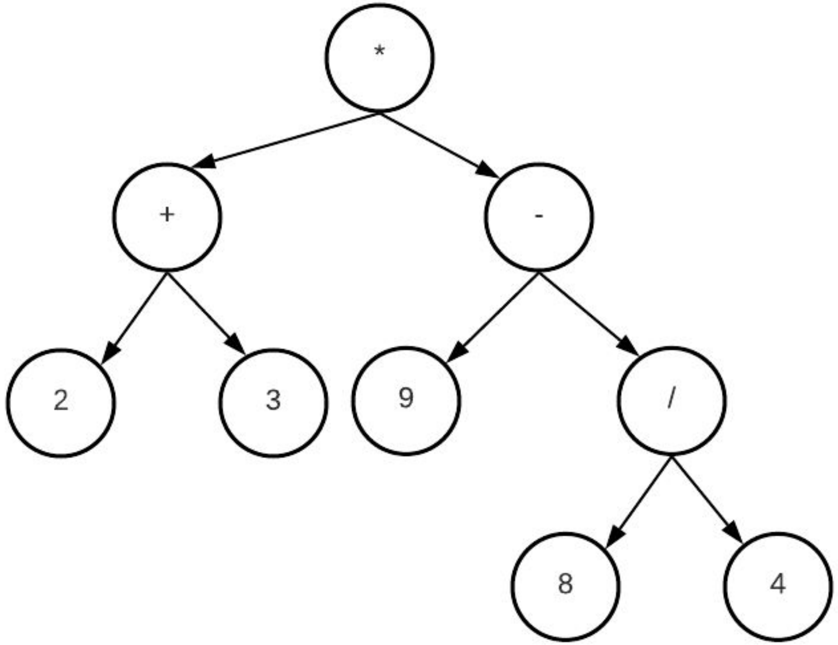
```
void print_postorder(Node* current) {
    if(!current)
        return;
    print_postorder(current->left);
    print_postorder(current->right);
    cout << current->data << " ";
}
```



Print Expression Tree: (2 + 3) * (9 - 8 / 4)

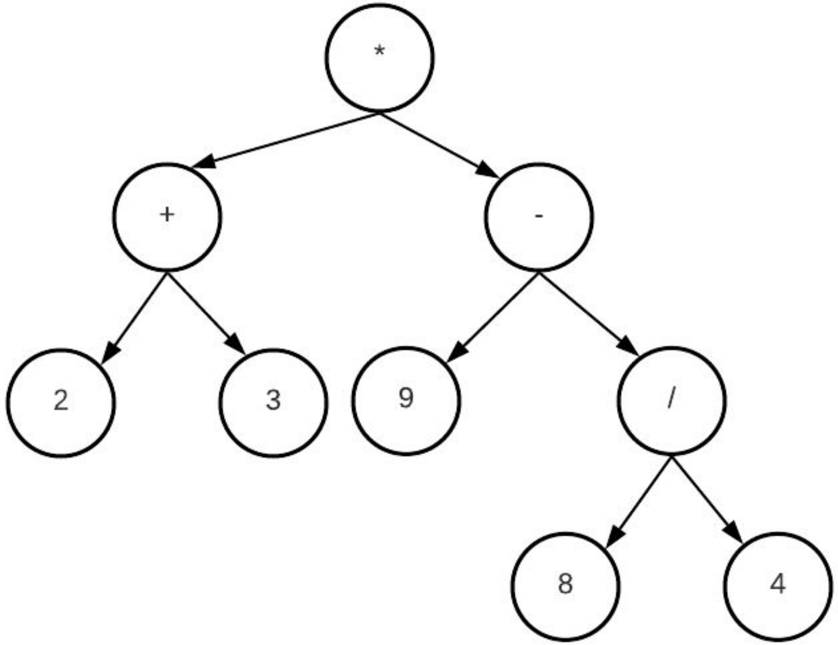


```
Node* plus = new Node('+');
plus->left = new Node('2');
plus->right = new Node('3');

Node* div = new Node('/');
div->left = new Node('8');
div->right = new Node('4');

Node* minus = new Node('-');
minus->left = new Node('9');
minus->right = div;

Node* multiply = new Node('*');
multiply->left = plus;
multiply->right = minus;
```



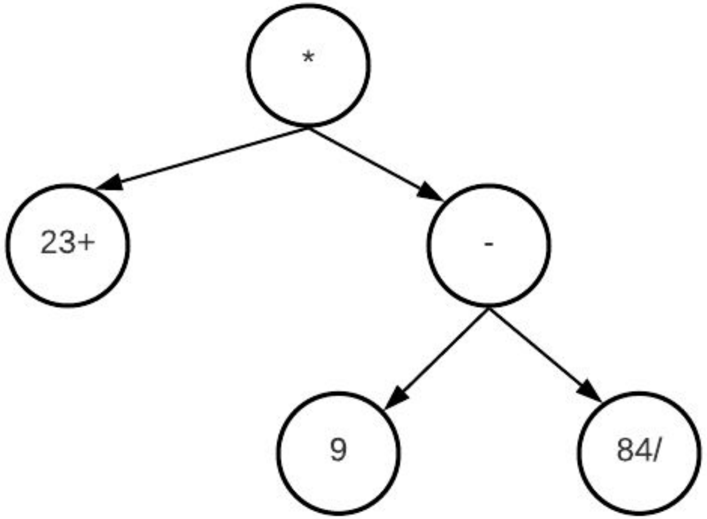
Print Expression Tree: (2 + 3) * (9 - 8 / 4)

- What is the postfix expression for:
- + subtree? 2 3 +
- / subtree? 8 4 /



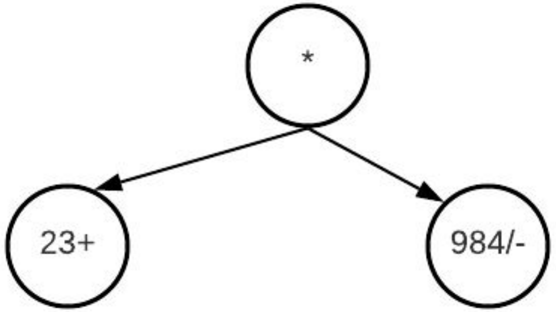
Print Expression Tree: (2 + 3) * (9 - 8 / 4)

- - subtree?
  - Left = 9
  - Right = 8 4 /
  - Value = -
  - Total: 9 **8 4 /** -



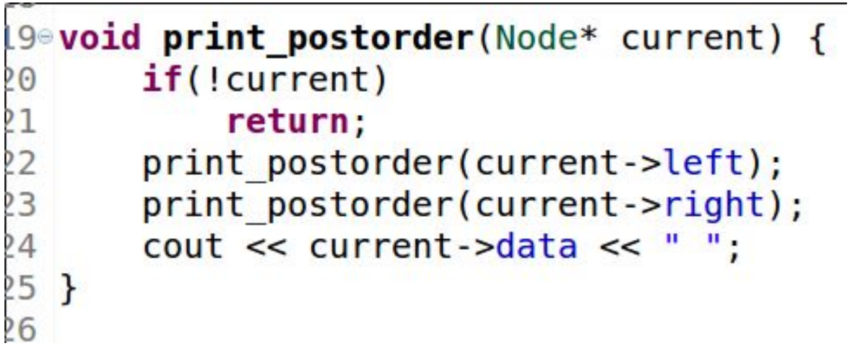
Print Expression Tree: (2 + 3) * (9 - 8 / 4)

- * subtree?
  - Left = 2 3 +
  - Right = 8 4 /
  - Value = *
  - Total: 2 3 +   9 8 4 / -   *

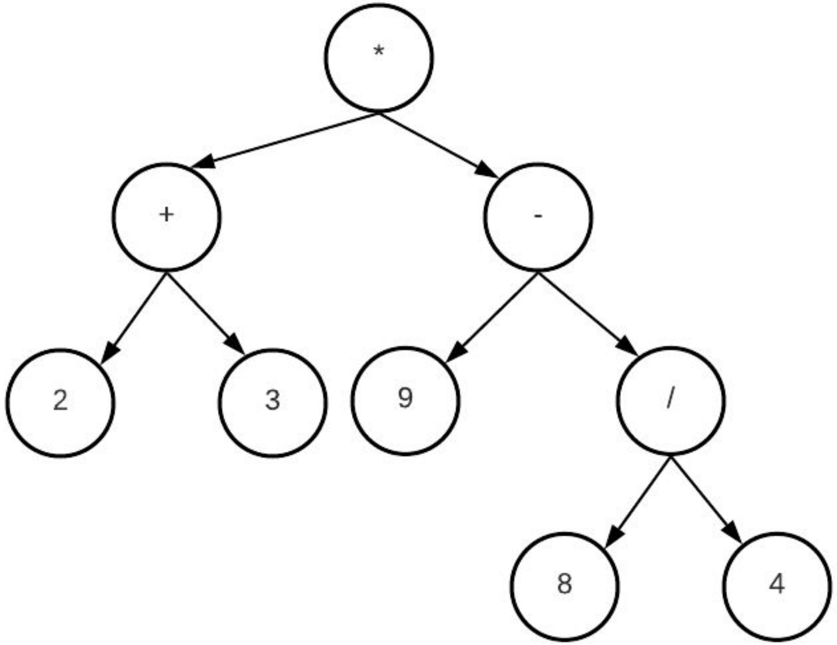


Print Expression Tree: (2 + 3) * (9 - 8 / 4)

- What:
  - + subtree ⇒ 2 3 +
  - / subtree ⇒ 8 4 /
  - - subtree ⇒ 9 8 4 /
  - * subtree ⇒ 2 3 +   9 8 4 / -   *



```
void print_postorder(Node* current) {
    if(!current)
        return;
    print_postorder(current->left);
    print_postorder(current->right);
    cout << current->data << " ";
}
```



Your turn

- Trace and understand very well!
- Clearing
  - We created a tree and recursively printed it!
  - But we need to free this memory!
  - Implement recursively: void clear(Node* current);

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."