

# COMP 250

## INTRODUCTORY SCIENCE

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat: edu\_assist\_pro

Week 11-3 Bin

Giulia Alberini, Fall 2020

Slides adapted from Michael Langer's

# WHAT ARE WE GOING TO DO IN THIS VIDEO?



- Binary Trees Assignment Project Exam Help
- Expression Tree <https://eduassistpro.github.io/>  
Add WeChat edu\_assist\_pro

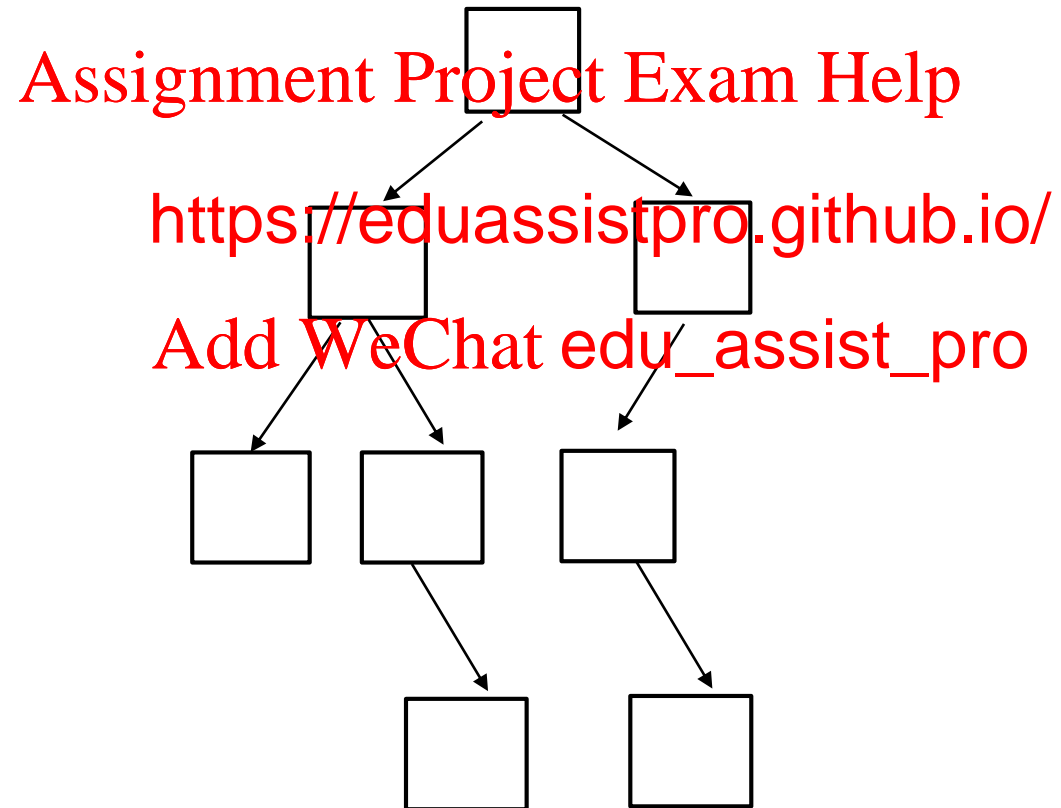
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

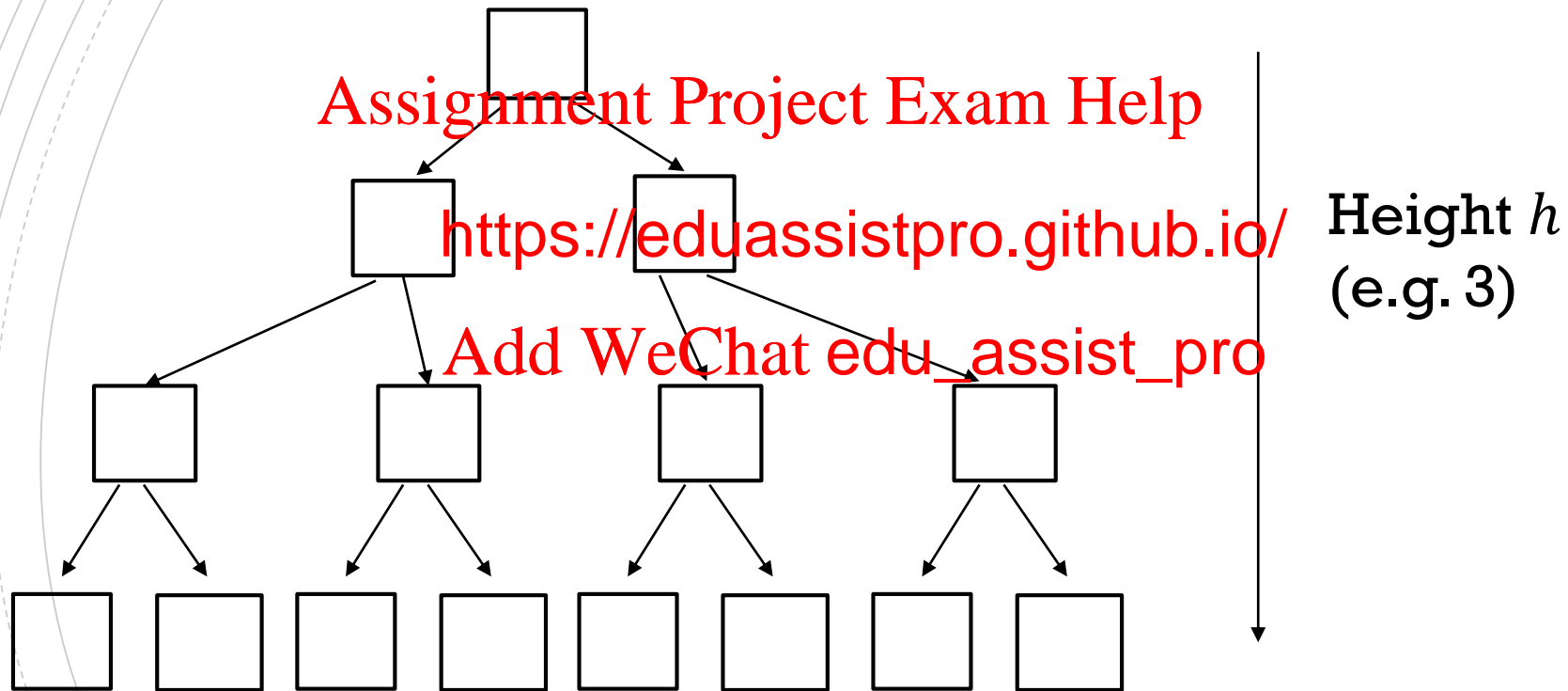
# BINARY TREES

Each node has at most 2 children!



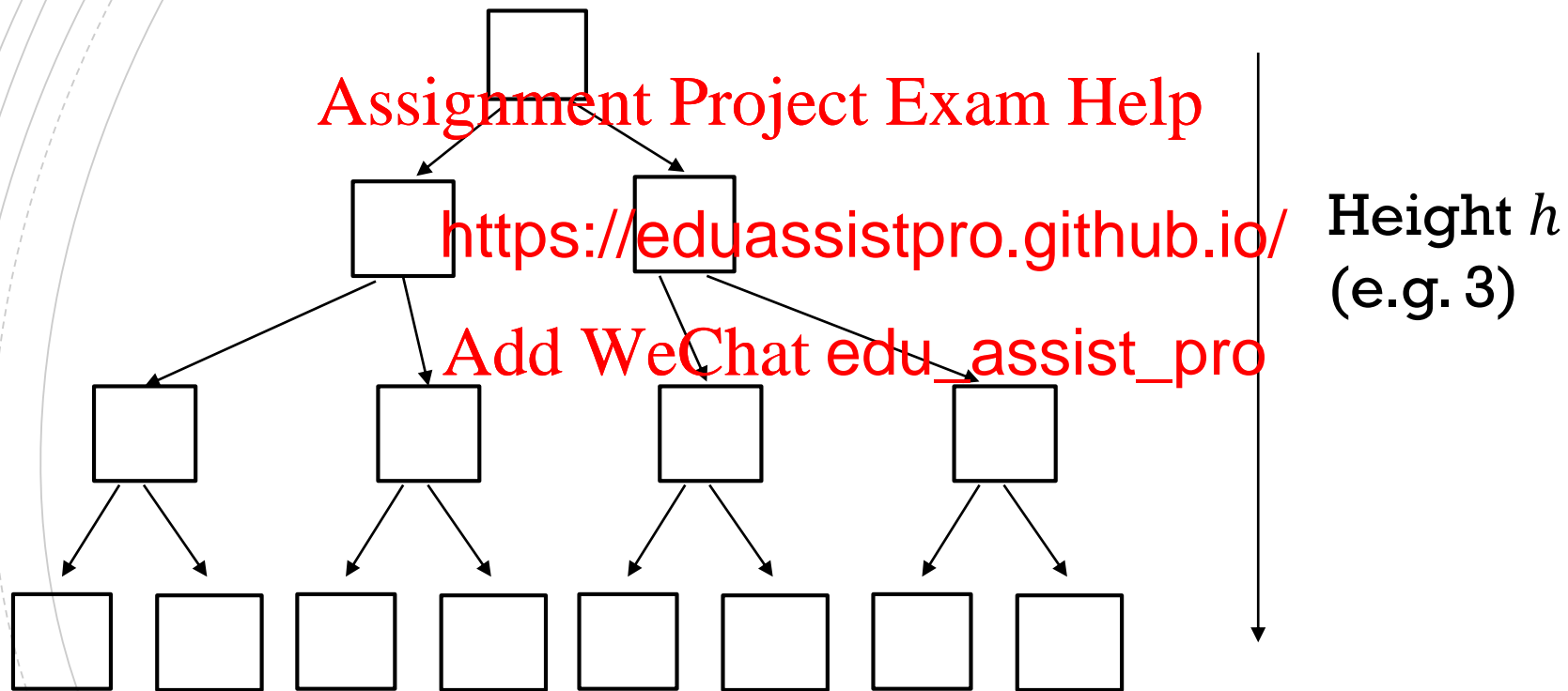
## BINARY TREES

Q: What is the maximum number of nodes  $n$  in a binary tree of height  $h$ ?



## BINARY TREES

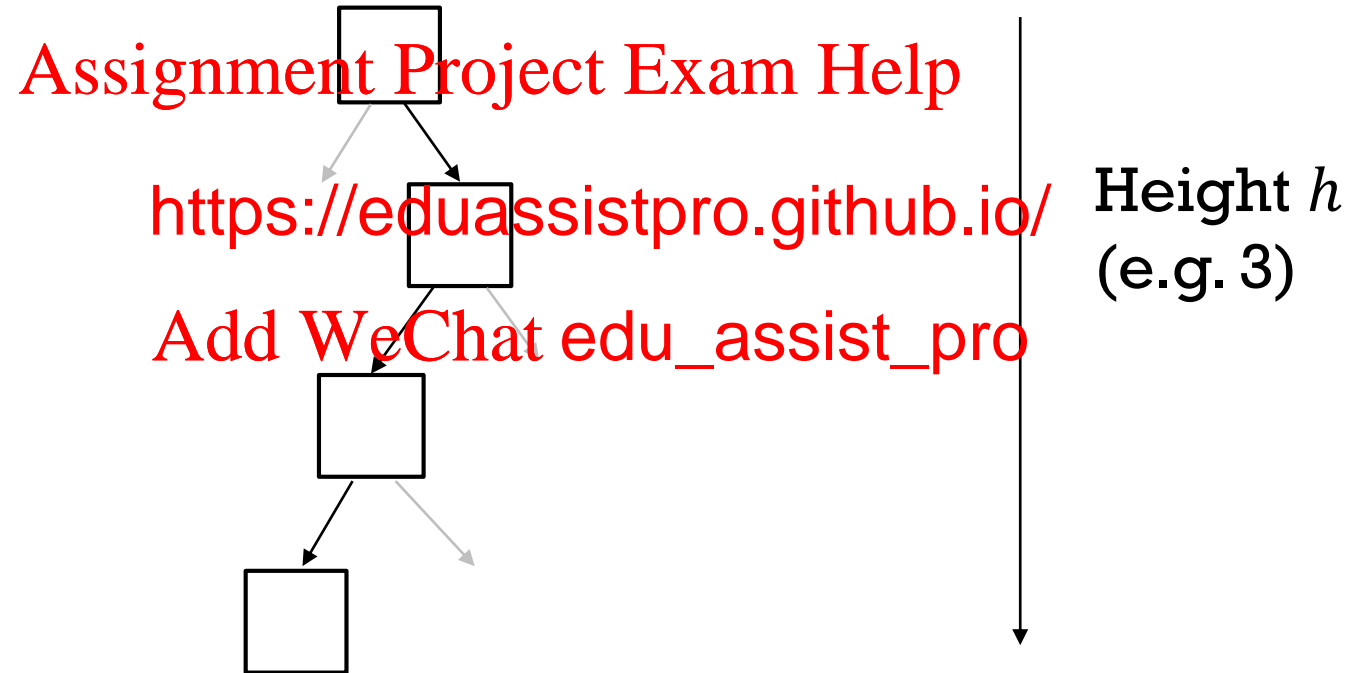
Q: What is the maximum number of nodes  $n$  in a binary tree of height  $h$ ?



A: 
$$n = 1 + 2 + 4 + \dots + 2^h = 2^{h+1} - 1$$

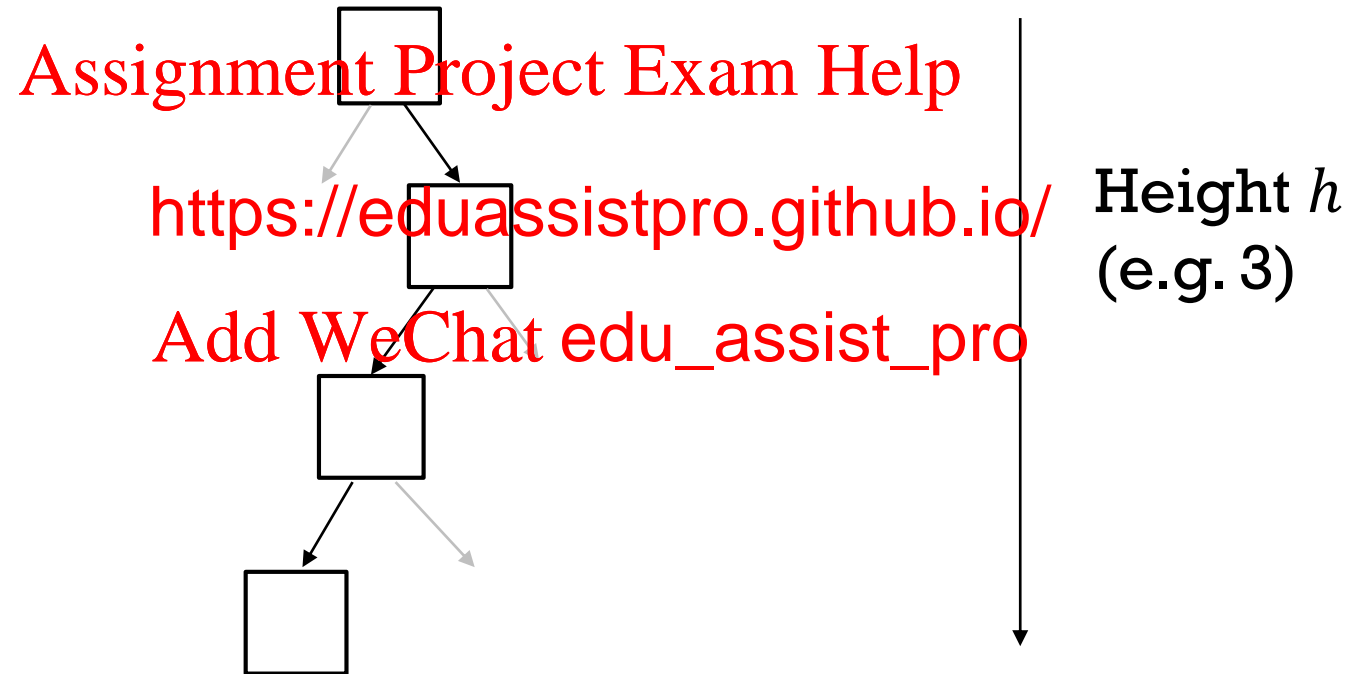
## BINARY TREES

Q: What is the minimum number of nodes  $n$  in a binary tree of height  $h$ ?



## BINARY TREES

Q: What is the minimum number of nodes  $n$  in a binary tree of height  $h$ ?



A:  $n = h + 1$



## BINARY TREES - IMPLEMENTATION

```
class BTree<T>{
```

```
    BTreeNode<T>    root;
```

**Assignment Project Exam Help**

**<https://eduassistpro.github.io/>**

```
class
```

```
    T element;
```

```
    BTreeNode<T> leftchild;
```

```
    BTreeNode<T> rightchild;
```

```
        :
```

```
    }
```

```
}
```

**Add WeChat edu\_assist\_pro**

# BINARY TREE TRAVERSAL (DEPTH FIRST)

Rooted Tree  
(last lecture)

Binary Tree

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

```
depthFirst(root) {  
  if (root is not empty) {  
    visit root  
    for each child of root  
      depthFirst( child )  
  }  
}
```

# BINARY TREE TRAVERSAL (DEPTH FIRST)

Rooted Tree  
(last lecture)

Binary Tree

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

```
preorder (root) {  
    if (root is not empty) {  
        visit root  
        for each child of root  
            preorder ( child )  
    }  
}
```

```
rBT (root) {  
    if (root is not empty) {  
        sit root  
        preorderBT (root.left)  
        preorderBT (root.right)  
    }  
}
```

## BINARY TREE TRAVERSAL (DEPTH FIRST)

```
preorderBT (root) {  
    if (root is not empty) {  
        visit root  
        preorderBT (root.left)  
        preorderBT (root.  
    }  
}
```

```
postorderBT (root) {
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

## BINARY TREE TRAVERSAL (DEPTH FIRST)

```
preorderBT (root) {  
    if (root is not empty) {  
        visit root  
        preorderBT (root.left)  
        preorderBT (root.  
    }  
}
```

```
postorderBT (root) {  
    if (root is not empty) {  
        postorderBT (root.left)  
        postorderBT (root.right)  
        visit root  
    }  
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# BINARY TREE TRAVERSAL (DEPTH FIRST)

```
preorderBT (root) {
    if (root is not empty) {
        visit root
        preorderBT (root.left)
        preorderBT (root.right)
    }
}
```

```
postorderBT (root) {
    if (root is not empty) {
        postorderBT (root.left)
        postorderBT (root.right)
        visit root
    }
}
```

# Assignment Project Exam Help

<https://eduassistpro.github.io/>

# Add WeChat edu\_assist\_pro

```
inorderBT (root) {
```

}

## BINARY TREE TRAVERSAL (DEPTH FIRST)

```
preorderBT (root) {  
    if (root is not empty) {  
        visit root  
        preorderBT (root.left)  
        preorderBT (root.  
    }  
}
```

```
postorderBT (root) {  
    if (root is not empty) {  
        postorderBT (root.left)  
        postorderBT (root.right)  
        visit root  
    }  
}
```

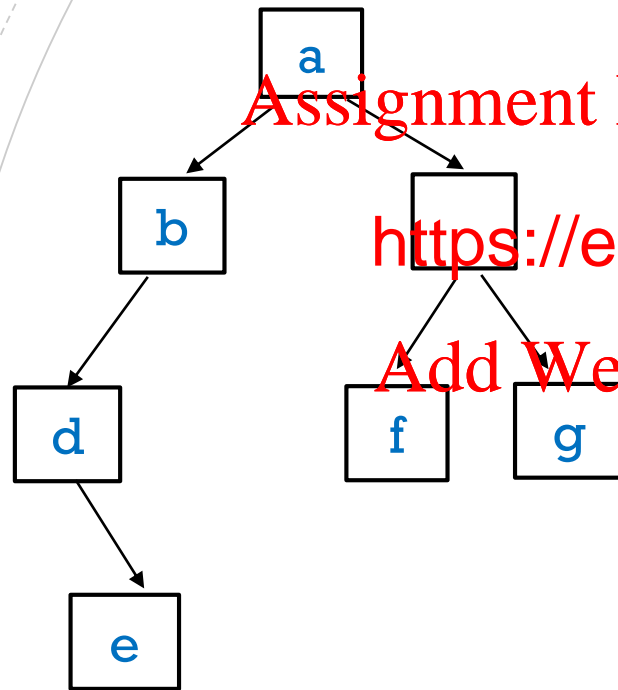
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

```
inorderBT (root) {  
    if (root is not empty) {  
        inorderBT (root.left)  
        visit root  
        inorderBT (root.right)  
    }  
}
```

## EXAMPLE



Assignment Project Exam Help

der:

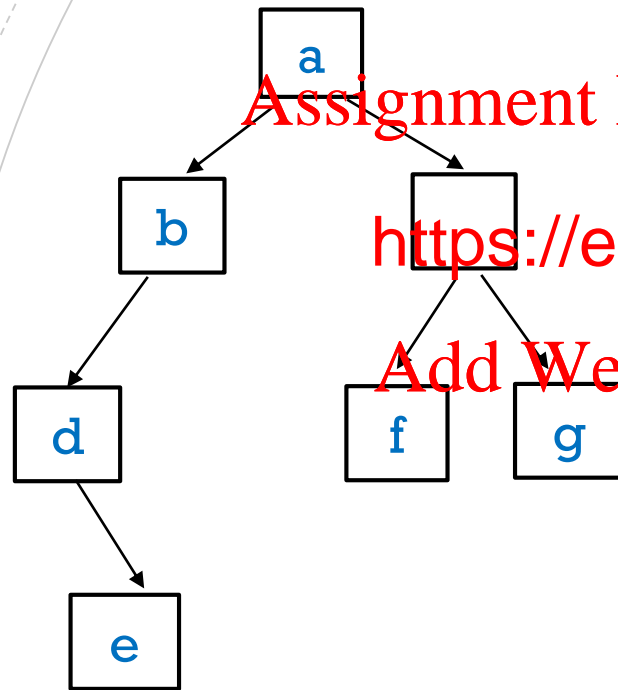
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

Post order:



## EXAMPLE



Assignment Project Exam Help

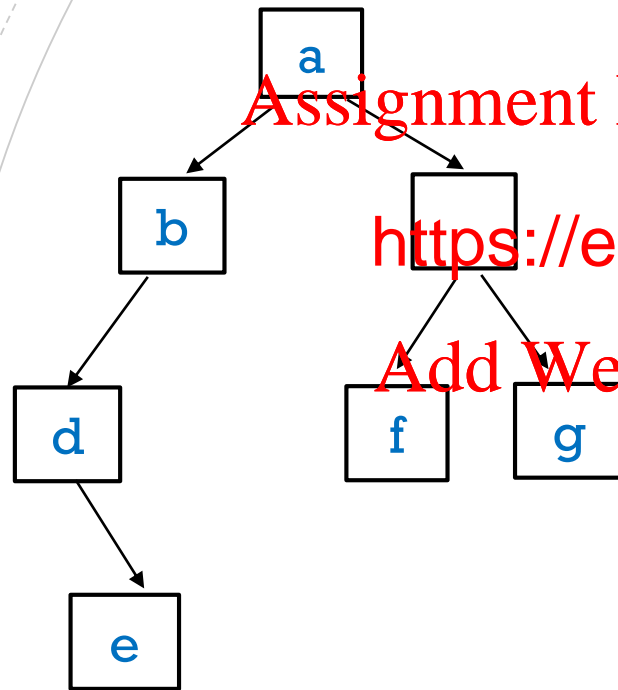
der: a b d e c f g

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

Post order:

## EXAMPLE



Assignment Project Exam Help

<https://eduassistpro.github.io/>

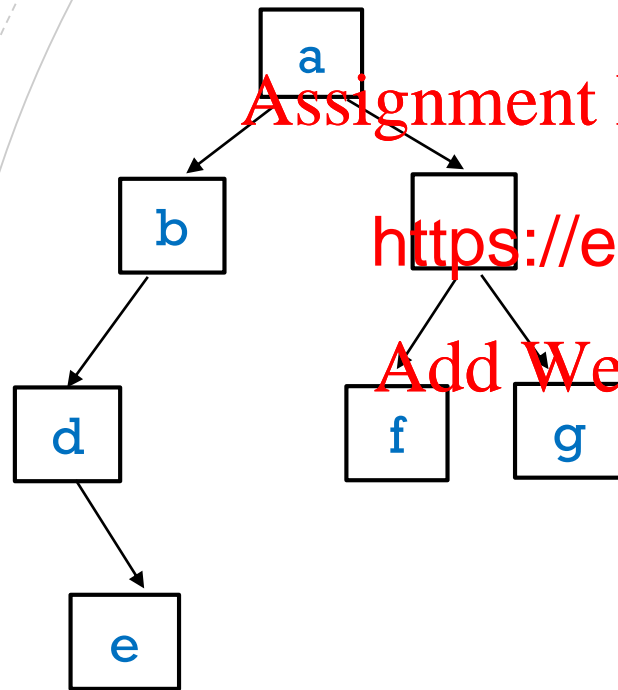
Add WeChat edu\_assist\_pro

der: a b d e c f g

d e b a f c g

Post order:

## EXAMPLE



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

der: a b d e c f g

d e b a f c g

Post order: e d b f g c a

Assignment Project Exam Help  
EX <https://eduassistpro.github.io/> EES  
Add WeChat edu\_assist\_pro

# EXPRESSION TREES

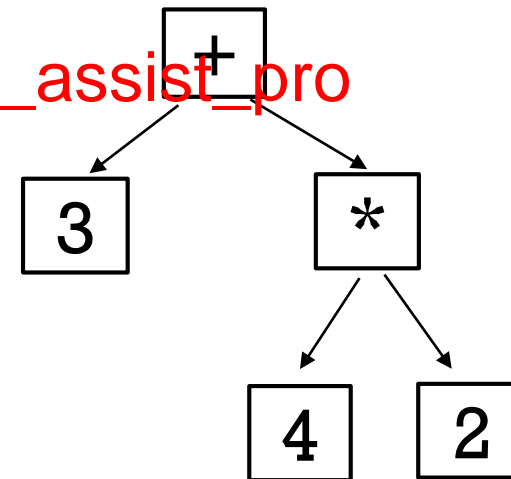
e.g.  $3 + 4 * 2$

Assignment Project Exam Help

$3 + (4 * 2)$

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



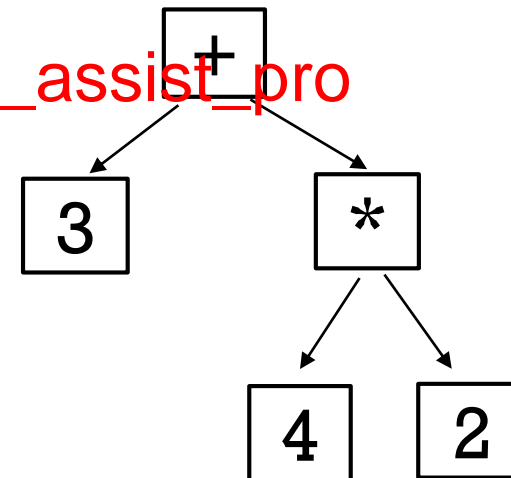
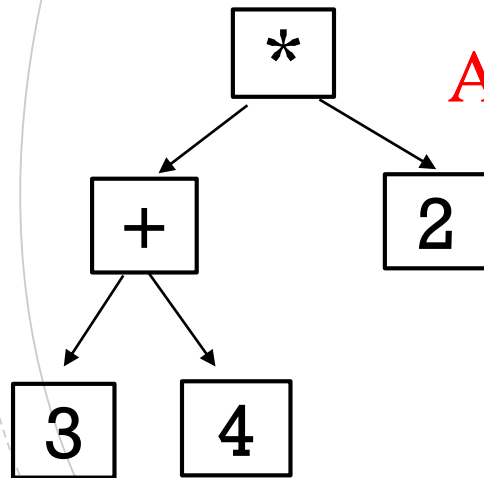
# EXPRESSION TREES

e.g.  $3 + 4 * 2$

$(3 + 4) * 2$        $3 + (4 * 2)$

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



My Windows calculator says

$$3 + 4 * 2 = 14.$$

Assignment Project Exam Help

<https://eduassistpro.github.io/> + 4) \* 2 = 14.

Add WeChat edu\_assist\_pro

if I google “3+4\*2”, I get 11.

$$3 + (4*2) = 11.$$

# EXPRESSIONS

We can make expressions using binary operators  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $^$

e.g.  $a - b / c + d * e ^ f ^ g$

Assignment Project Exam Help

$^$  is exponentiation: <https://eduassistpro.github.io/>

We don't consider unary operators  $e$   $3 + (-4)$

Add WeChat edu\_assist\_pro

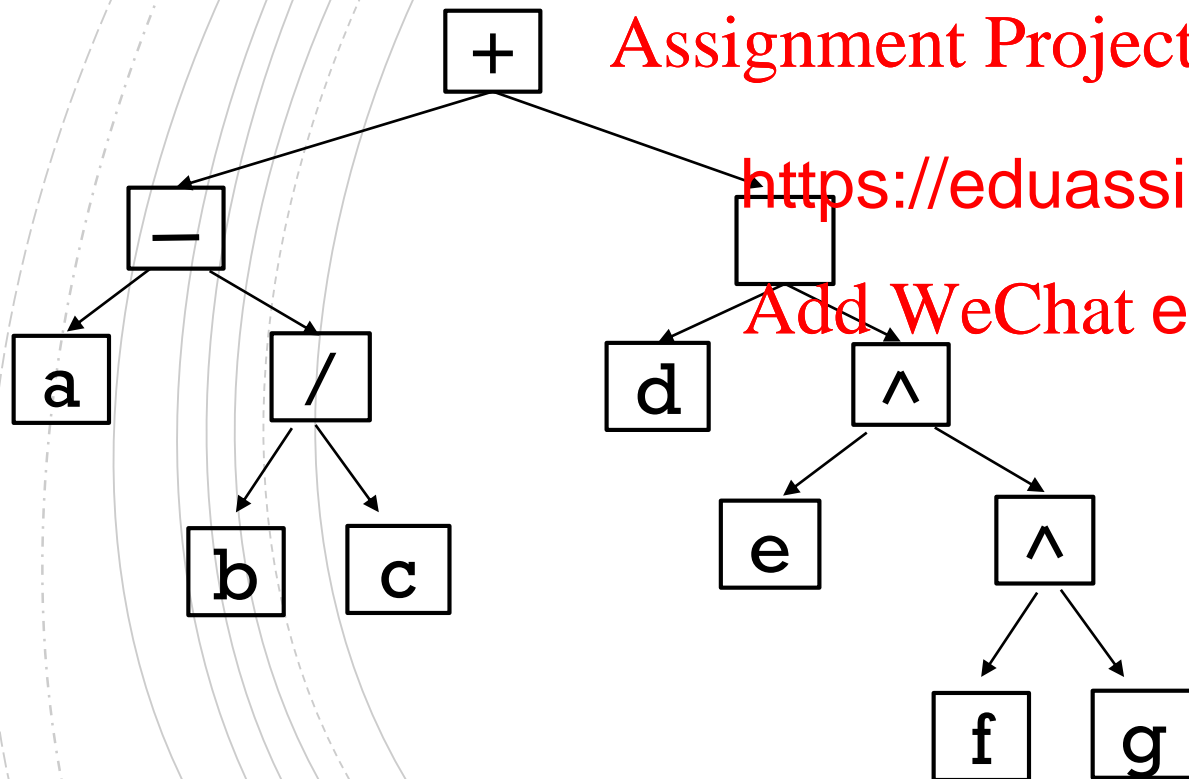
Operator precedence ordering makes brackets unnecessary.

$$(a - (b / c)) + (d * (e ^ (f ^ g)))$$



## EXPRESSION TREE

$$a - b / c + d * e ^ f ^ g \equiv (a - (b / c)) + (d * (e ^ (f ^ g)))$$



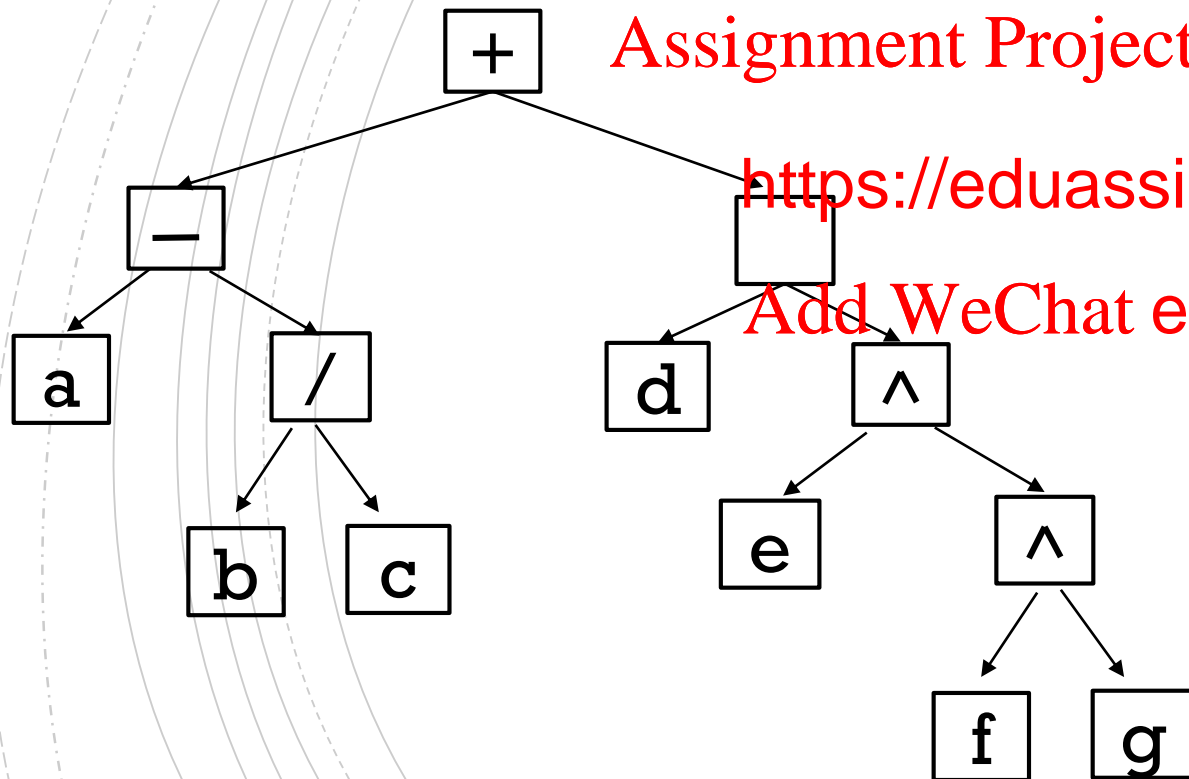
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

1 nodes are operators.  
s are operands.

# EXPRESSION TREE



Assignment Project Exam Help

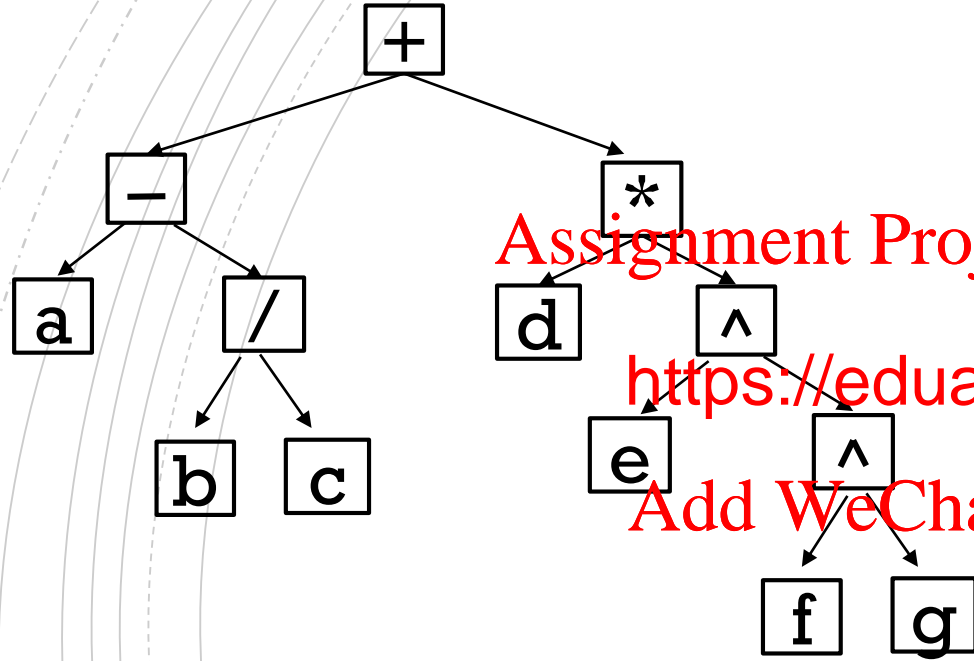
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

An expression tree can be a way of *thinking about* the ordering of operations used evaluating an expression.

But to be concrete, *let's assume we have a binary tree data structure.*

## EXPRESSION TREE – TRAVERSAL



If we traverse an expression tree,  
and *print out* the node label, what  
expression printed out?

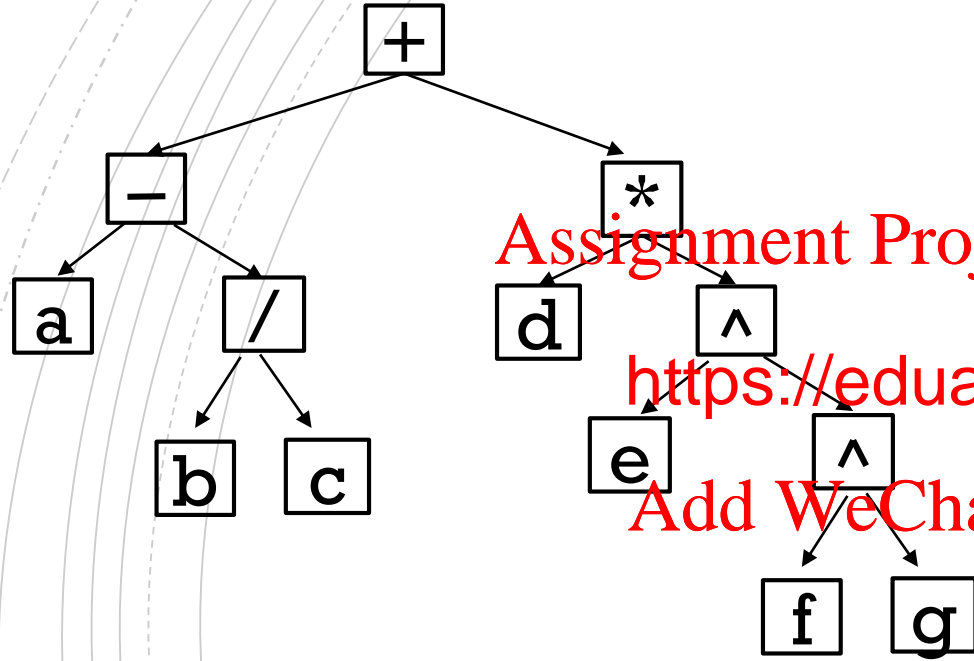
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

preorder traversal gives:

## EXPRESSION TREE – TRAVERSAL



If we traverse an expression tree,  
and *print* out the node label, what  
expression printed out?

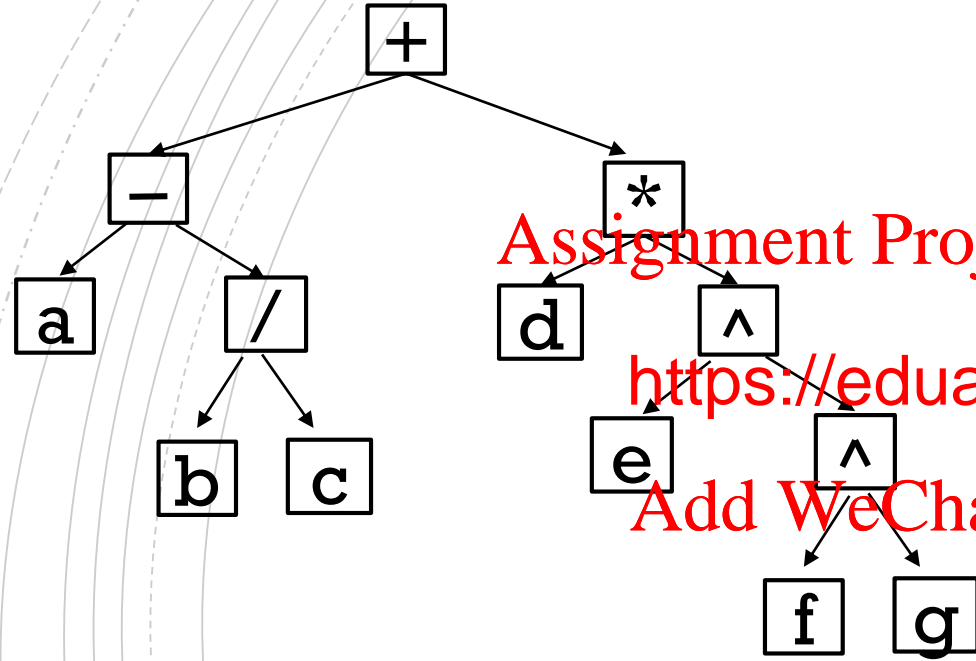
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

preorder traversal gives:  $+ - a / b c * d ^ e ^ f g$

## EXPRESSION TREE – TRAVERSAL



If we traverse an expression tree,  
and *print* out the node label, what  
expression printed out?

Assignment Project Exam Help

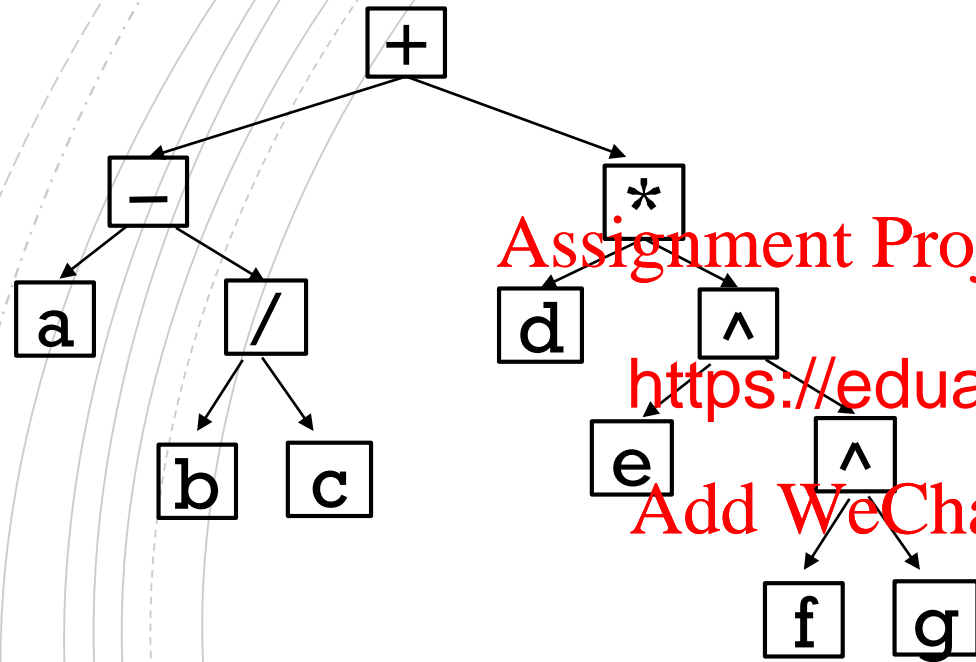
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

preorder traversal gives:  $+ - a / b c * d ^ e ^ f g$

inorder traversal gives:

## EXPRESSION TREE – TRAVERSAL



If we traverse an expression tree,  
and *print* out the node label, what  
expression printed out?

Assignment Project Exam Help

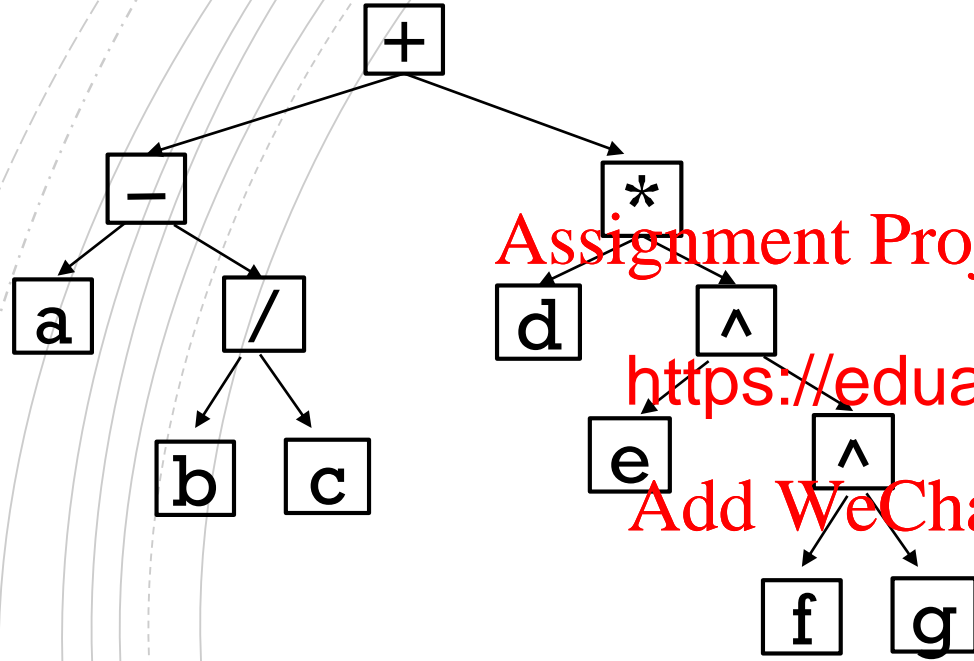
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

preorder traversal gives:  $+ - a / b c * d ^ e ^ f g$

inorder traversal gives:  $a - b / c + d * e ^ f ^ g$

## EXPRESSION TREE – TRAVERSAL



If we traverse an expression tree,  
and *print* out the node label, what  
expression printed out?

Assignment Project Exam Help

<https://eduassistpro.github.io/>

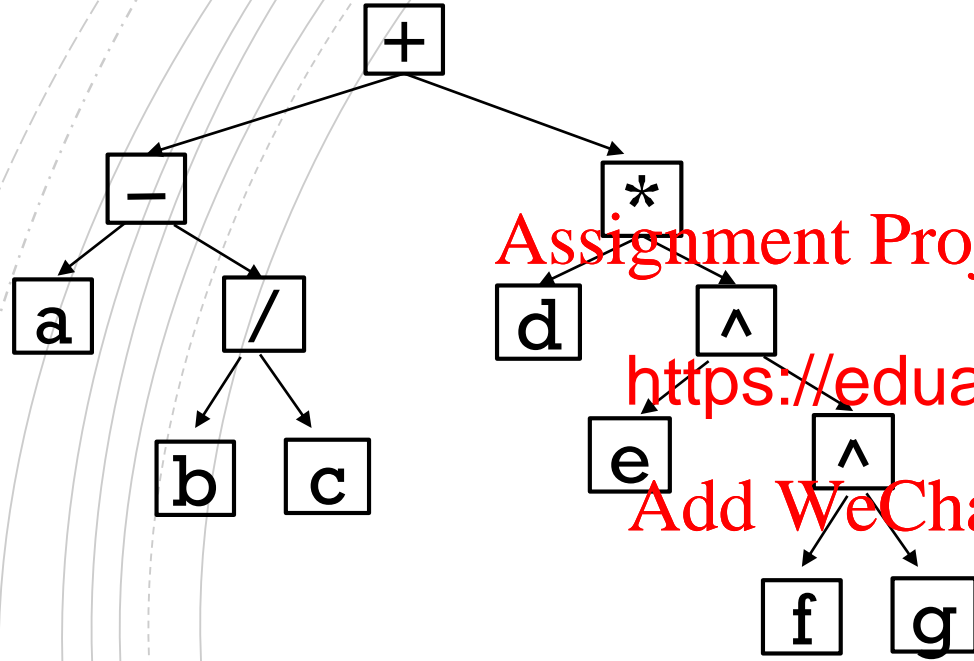
Add WeChat edu\_assist\_pro

preorder traversal gives:  $+ - a / b c * d ^ e ^ f g$

inorder traversal gives:  $a - b / c + d * e ^ f ^ g$

postorder traversal gives:

## EXPRESSION TREE – TRAVERSAL



If we traverse an expression tree,  
and *print* out the node label, what  
expression printed out?

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

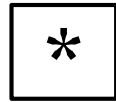
preorder traversal gives:  $+ - a / b c * d ^ e ^ f g$

inorder traversal gives:  $a - b / c + d * e ^ f ^ g$

postorder traversal gives:  $a b c / - d e f g ^ ^ * +$



# PREFIX, INFIX, POSTFIX EXPRESSIONS



Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

prefix:     \* a b

infix:       a \* b

postfix:     a b \*

## PREFIX, INFIX, POSTFIX EXPRESSIONS

**baseExp** = variable | integer

**op** = + | <https://eduassistpro.github.io/>

**preExp** = baseExp | op **preExp**

where | means 'or'

## PREFIX, INFIX, POSTFIX EXPRESSIONS

baseExp = variable | integer

Assignment Project Exam Help

op = + | <https://eduassistpro.github.io/>

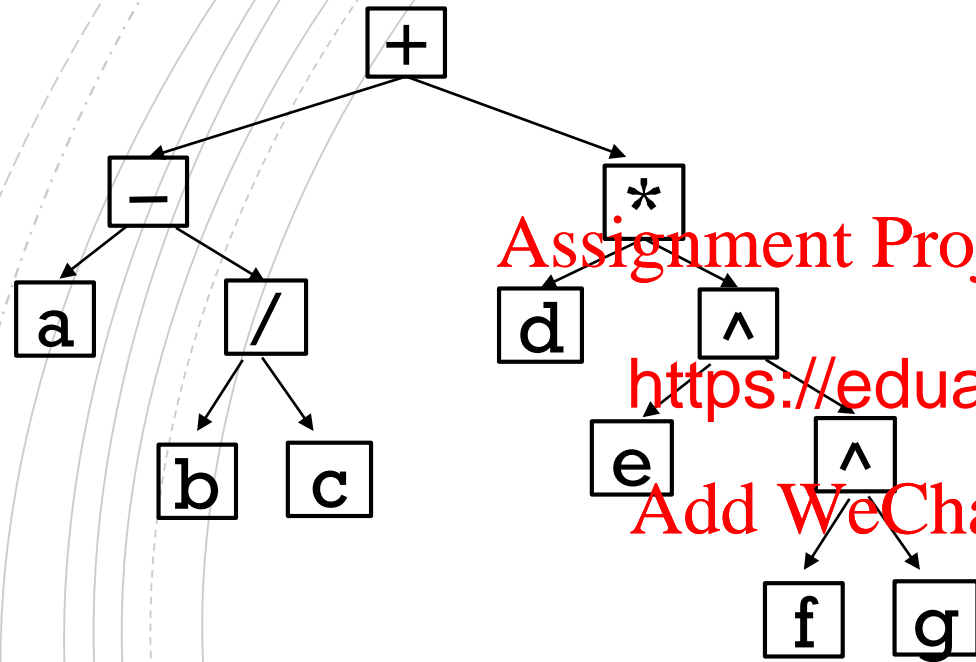
preExp = baseExp | op preExp

inExp = baseExp | inExp op inExp

postExp = baseExp | postExp postExp op

Use  
only  
one.

## EXPRESSION TREE – TRAVERSAL



If we traverse an expression tree,  
and *print* out the node label, what  
expression printed out?

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

preorder traversal gives **prefix expression**:  $+ - a / b c * d ^ e ^ f g$

inorder traversal gives **infix expression**:  $a - b / c + d * e ^ f ^ g$

postorder traversal gives **postfix expression**:  $a b c / - d e f g ^ ^ * +$

## TERMINOLOGY

- **Prefix** expressions called “Polish Notation”  
(after Polish logician Jan Lucasewicz 1920's)  
<https://eduassistpro.github.io/>  
**Assignment Project Exam Help**
- **Postfix** expressions  
<https://eduassistpro.github.io/>  
**Add WeChat edu\_assist\_pro**  
lish notation” (RPN)

Some calculators (esp. Hewlett Packard) require users to input expressions using RPN.

# TERMINOLOGY

- **Prefix** expressions called “Polish Notation”  
(after Polish logician Jan Lucasewicz 1920's)

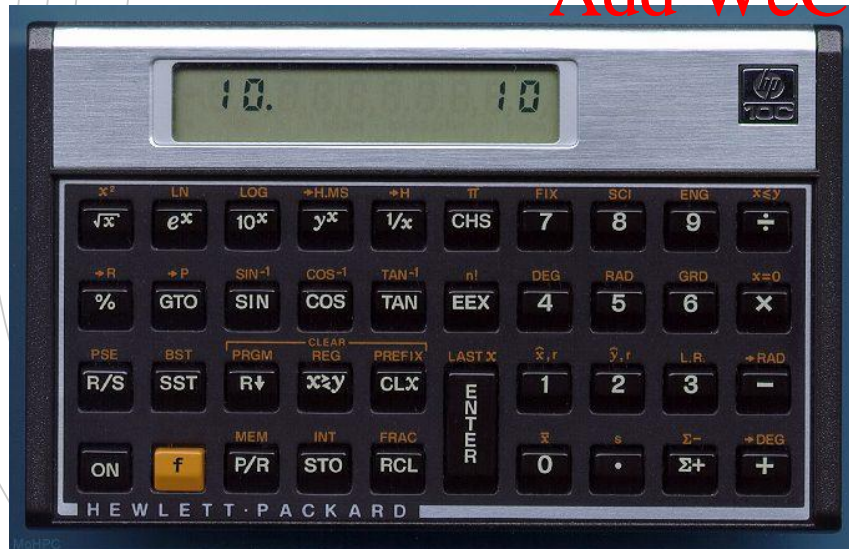
Assignment Project Exam Help

- **Postfix** expressions

<https://eduassistpro.github.io/>  
lish notation” (RPN)

Add WeChat edu\_assist\_pro

\* 4 + 3 :



5 <enter>  
4 <enter>  
\* <enter>  
3 <enter>  
+ <enter>

No “=” symbol on keyboard.

An orange paint roller with a red handle, positioned horizontally. The roller is partially filled with orange paint, and there are orange paint splatters and drips around it. The text "Coming Soon" is written in white on the orange background of the roller.

# Coming Soon

## Assignment Project Exam Help

In the next

- Binary S <https://eduassistpro.github.io/>
- Heaps Add WeChat edu\_assist\_pro