RIZAL TECHNOLOGICAL UNIVERSITY
COLLEGE OF ENGINEERING ARCHITECTURE AND TECHNOLOGY

DATA STRUCTURE AND ALGORITHM

TREES

- □ What is Tree?
- □ Binary Trees
- □ Types of Binary Trees
- □ Binary Tree Traversal
- □ Properties of Binary Trees

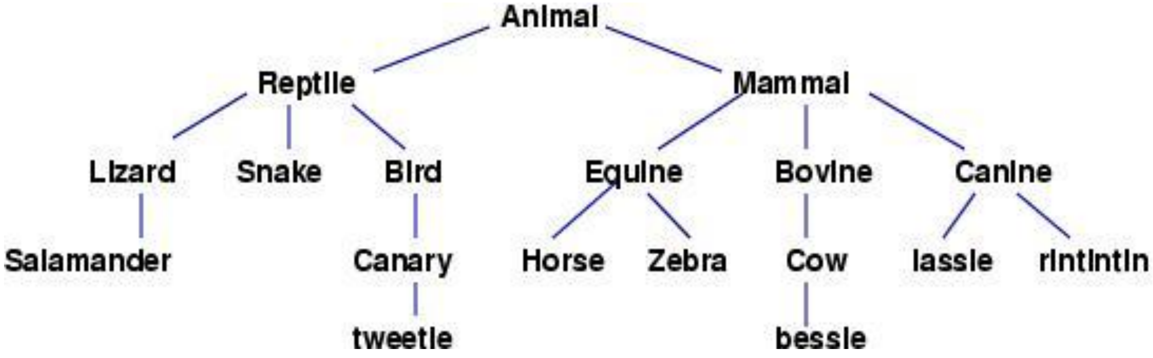
Objectives:

- □ Identify the terms in determining a tree,
- □ Differentiate binary trees from expressions trees
- □ Differentiate the operations that can be performed in a binary tree
- □ Enumerate the types of binary tree traversal

TREES

- root is on the top and the leaves are at a bottom
- display data items in a hierarchical structure
- organize information in database systems
- also work as ordered array and a linked list

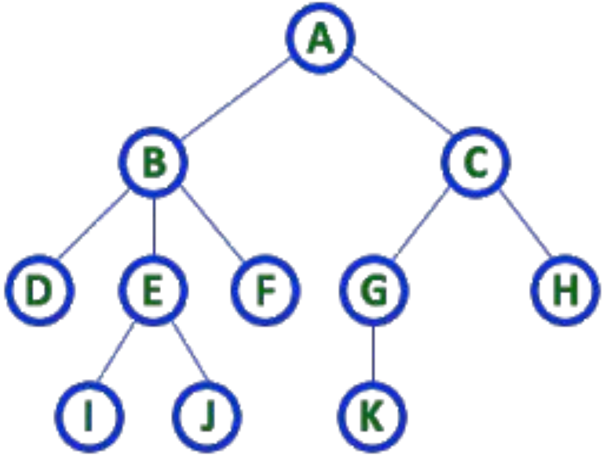
Example:



TREES

- a tree is composed of node connected by edges or lines

Example:



TREE with 11 nodes and 10 edges

- In any tree with 'N' nodes there will be maximum of 'N-1' edges

- In a tree every individual element is called as 'NODE'

TREES : Basic Terminologies

- **Node.** what trees composed of connected by edges or lines that contain a value or represent a data structure.

- **Root Node.** the node at the topmost of the tree and where the tree commonly begins

- **Edges.Lines.Paths.** lines connecting to nodes describing their relationship

- **Leaf Nodes.** nodes that are at the bottommost level of the tree and so, they have no children

- **Internal Nodes/Child Nodes.** nodes that are inside a tree or the nodes below a given node which also known as sibling nodes

TREES : Basic Terminologies

- **Subtree.** It is viewed as a complete tree itself, which consists of children and its children's children and so on

- **Visiting.** A node is said to be visited when a program control arrives at the node to perform operations such as checking the value of one of its data fields, or simply to display it.

- **Traversing.** To visit all the nodes in a specific order is called traverse or more commonly known as walking the tree, and the action is walk.

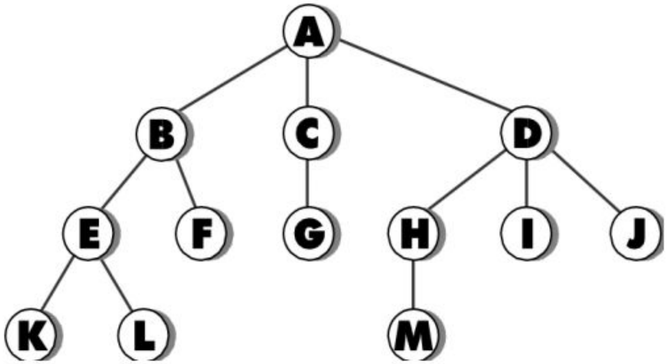
- **Levels.** These are generations that a tree has started at the root.

TREES : Basic Terminologies

- **Degree.** This refers to the number of nodes in a subtree.
- **Depth of the tree.** This pertains to the highest level of the tree.
- **Keys.** These represent the value inside a node represented in a circle.

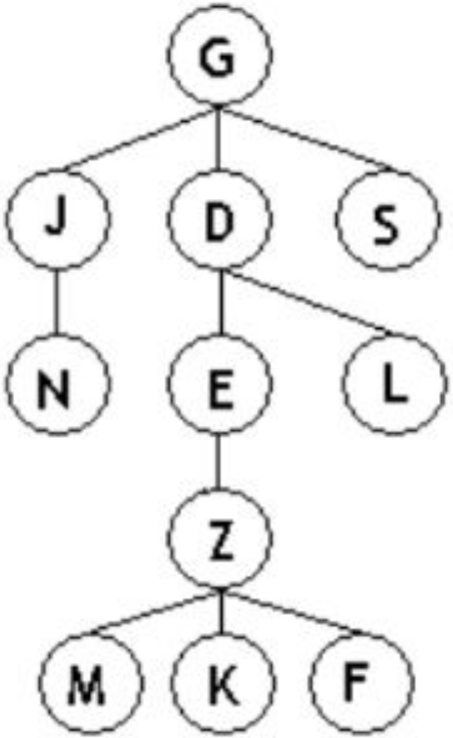
TREES

Example:



Depth of Tree.	4
Degree.	Degree of A is 3 Degree of B is 2 Degree of C is 1 Degree of K is 0 The maximum degree is 3
Root Node.	A
Internal Nodes.	B, C, D, E, F, G, H, I, J, K, L, M
Leaves or Leaf Nodes.	K, L, M, F, G, I, J
Levels.	4 Level 1 – A Level 2 – B, C, D Level 3 – E, F, G, H, I, J Level 4 – K, L, M
Subtree.	B, E, F C, G E, K, L D, H, I, J H, M

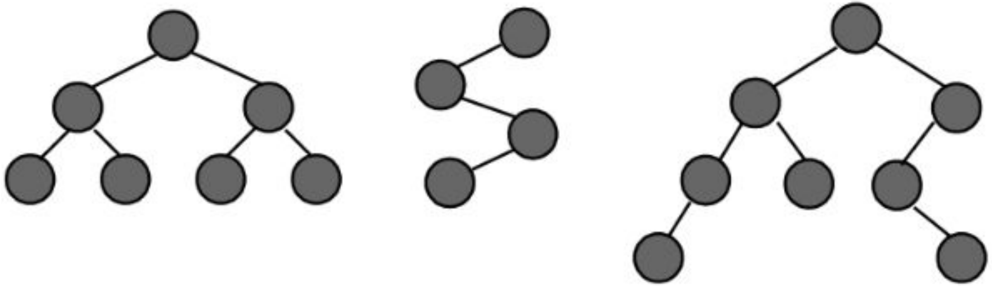
Exercise Problem



- Root Node
- Depth of the tree
- Nodes at level 4
- Degree of D
- Children of Z
- Total Nodes in the Tree
- Parent of F
- Leaf Nodes
- Child nodes
- Maximum degree of the tree

BINARY TREE

- if every node in a tree can have at most two children
  - left subtree
  - right subtree
- can contain only left or right subtree or no nodes at all, in which case it's a leaf

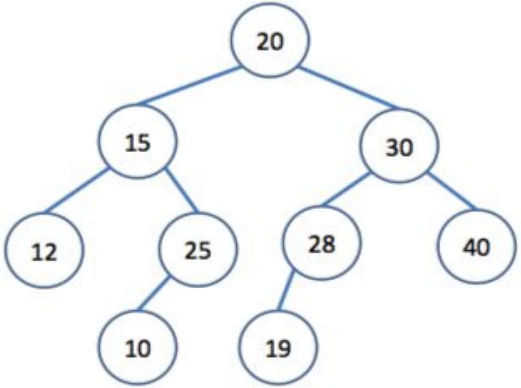


- two applications of binary trees:
  - binary expression tree
  - binary search tree

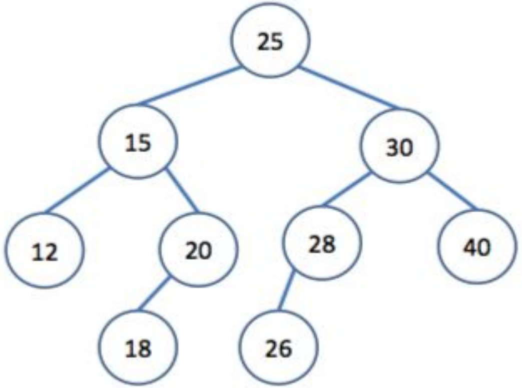
Binary Search Tree

- **E:\RTU\CPE\lessons\Data Structure and Algorithms\ppt\07 Trees.ppt**
- Hierarchical file structure in a computer system is one of the common trees
- Hierarchical file structure is not a binary tree
- Binary search trees are trees such that any node has a key which is no more than the key in its right child node and no less than the key in its left child node

Binary Search Tree



not a BST

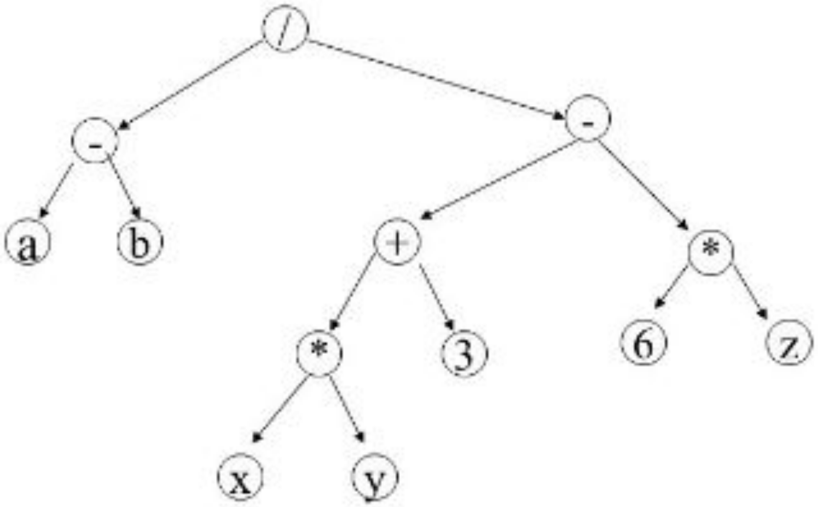


valid BST

Binary Expression Tree

- is a binary tree used in representing an arithmetic expression
- specifically combinations of operators, operands, and order of evaluation

Example: (a-b) / ((x*y+3)-(6*z))



Binary Expression Tree

Conducting an Expression Tree:

- Read the symbols one at a time
- If the symbol is an operand:

  - Create a one-node tree

  - Push a pointer to the node onto the stack
- If the symbol is an operator:

  - Pop two pointers from stack. These pointers represent the address of the root nodes of two trees, T1 and T2.

  - Form a new tree whose root is the operator and whose left and right children are T1 and T2, respectively.

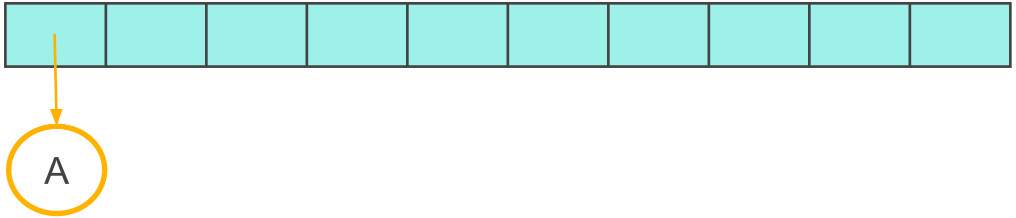
  - Push the pointer to the new tree onto the stack.
- Repeat steps 1 thru 3 until the last symbol has been read and processed.

Binary Expression Tree

Example Expression: AB+CDE+*/

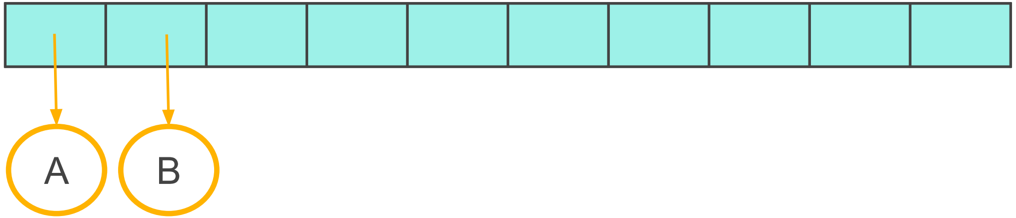
1. Read the first symbol: “A”

2. Create a one-node tree for “A” and push pointer to “A” onto a stack.



3. Read next symbol: “B”

4. Create a one node tree for “B” and push a pointer to “B” into the stack.

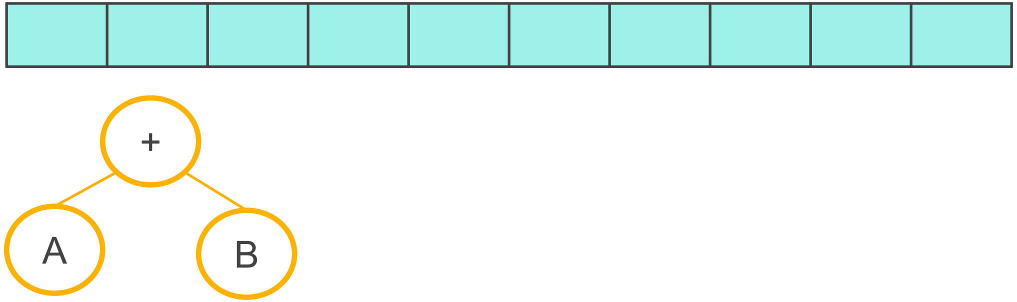


Binary Expression Tree

Example Expression: AB+CDE+*/

5. Read next symbol: “+”

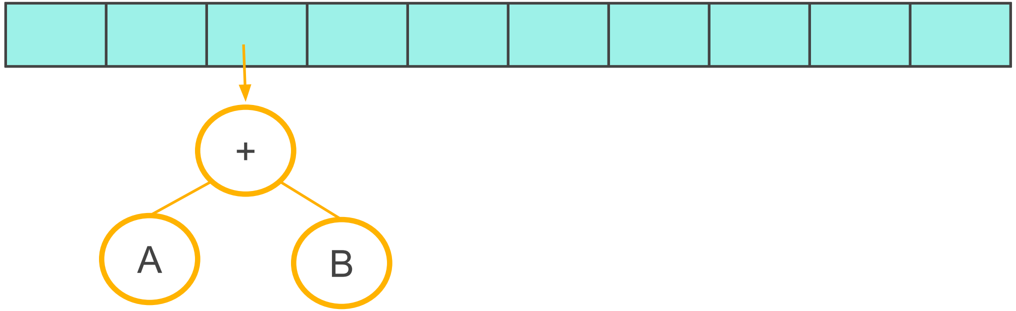
6. Pop two pointers from the stack: Create a tree whose root is “+”. Make “A” the left subtree and “B” the right subtree.



Binary Expression Tree

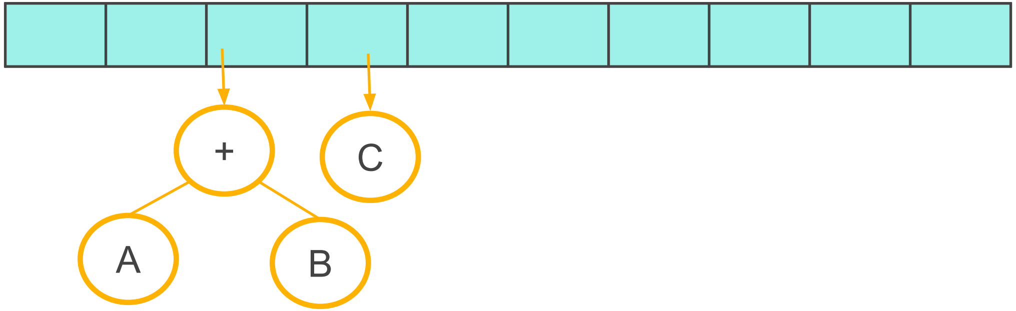
Example Expression: AB+CDE+*/

7. Push the pointer to the new tree into the stack.



8. Read next symbol: “C”

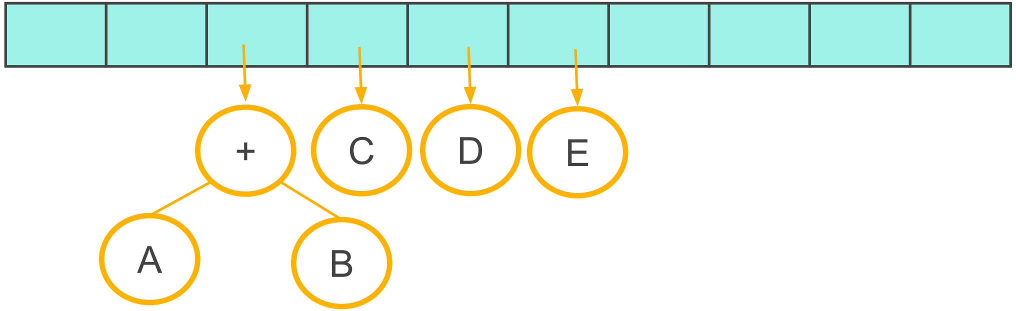
9. Create a one node tree for “C” and push a pointer to “C” into the stack.



Binary Expression Tree

Example Expression: AB+CDE+*/

10. Repeat the same process for the next two symbols: “D” and “E”.

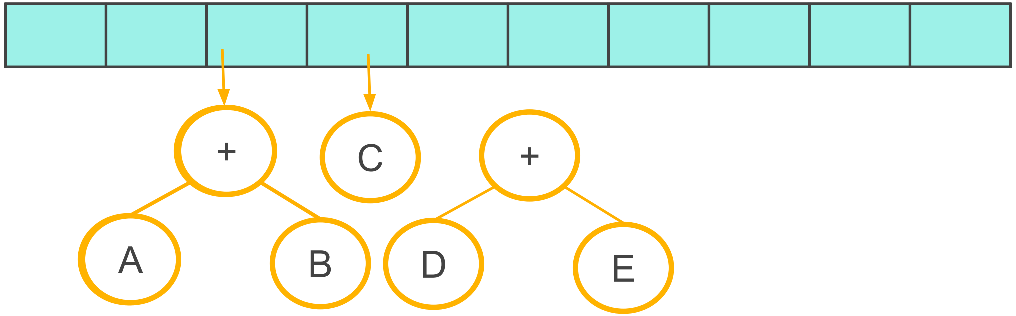


11. Read the next symbol: “+”.

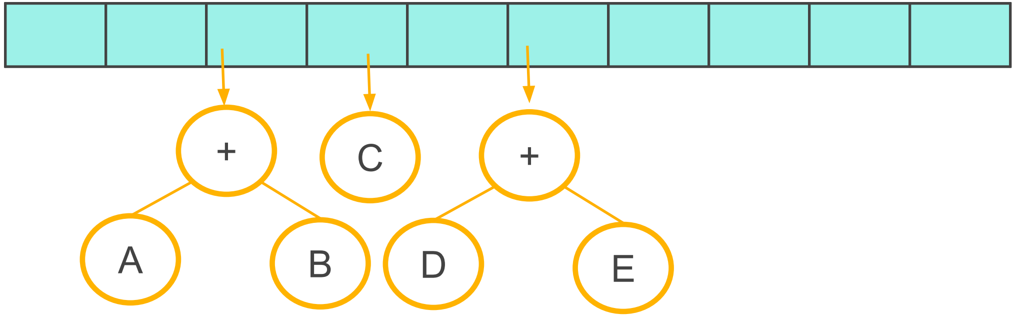
Binary Expression Tree

Example Expression: AB+CDE+*/

12. Pop two pointers from the stack: Create a tree whose root is "+". Make "D" the left subtree and "E" the right subtree.



13. Push the pointer to the new tree into the stack.

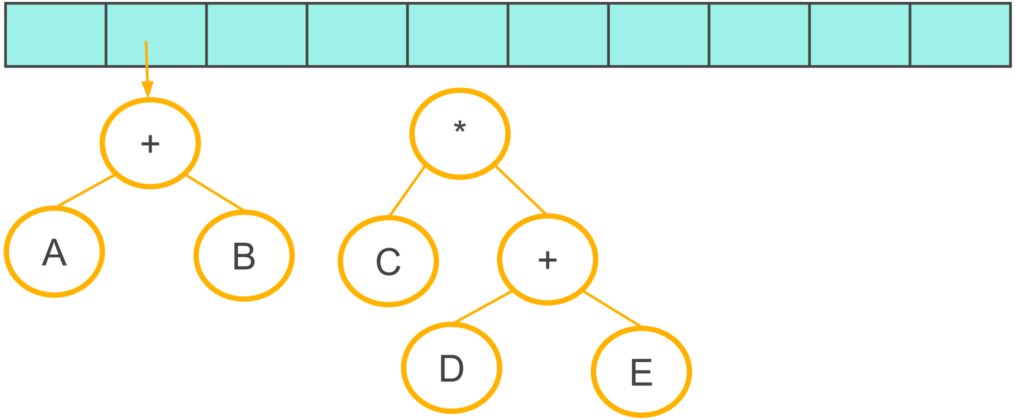


Binary Expression Tree

Example Expression: AB+CDE+*/

12. Read the next symbol: "∗"

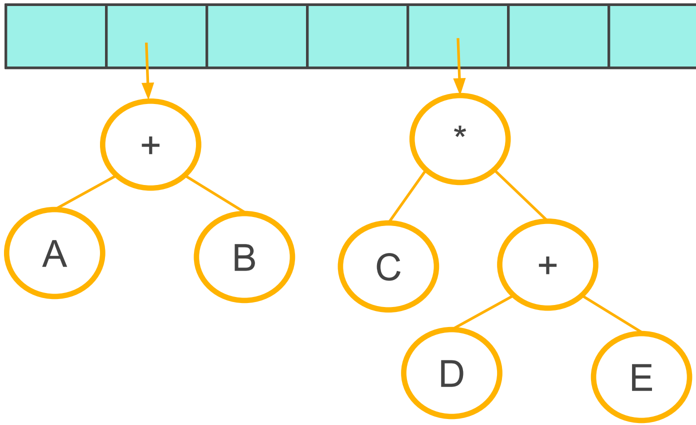
13. Pop two pointers from the stack: Create a tree whose root is "*". Make "C" the left subtree and the right subtree.



Binary Expression Tree

Example Expression: AB+CDE+*/

16. Push the pointer to the new tree into the stack.

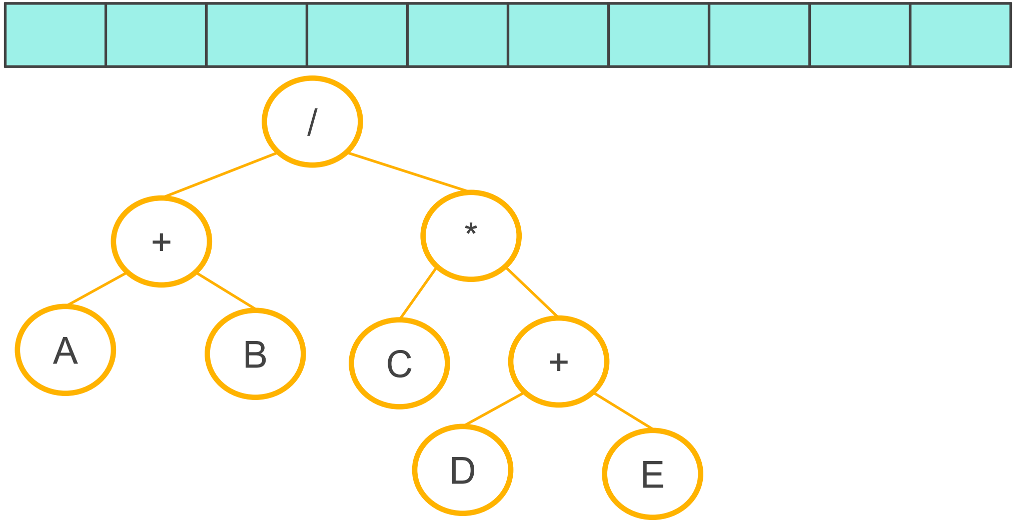


17. Read the next symbol: “/”

Binary Expression Tree

Example Expression: AB+CDE+*/

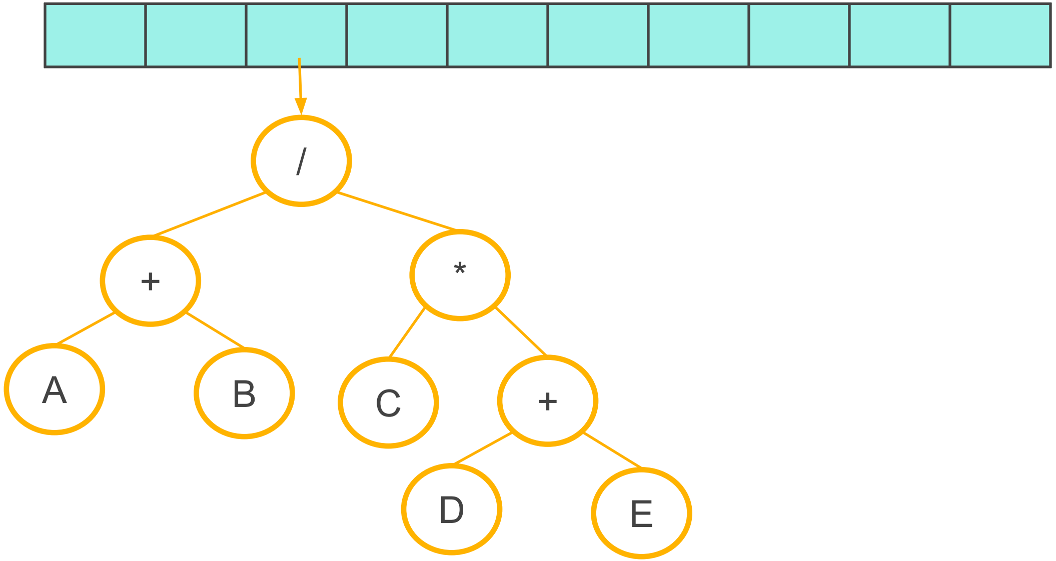
18. Pop two pointers from the stack: Create a tree whose root is “/”. Make the left subtree and the right subtree..



Binary Expression Tree

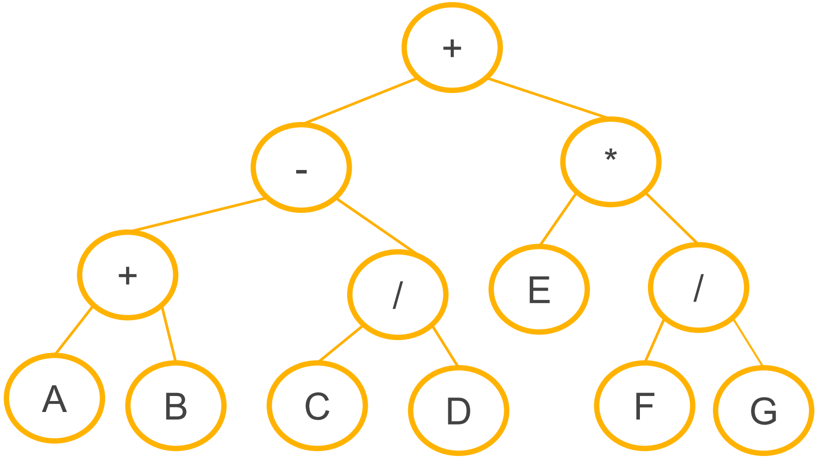
Example Expression: AB+CDE+*/

19. Push the pointer to the new tree into the stack.



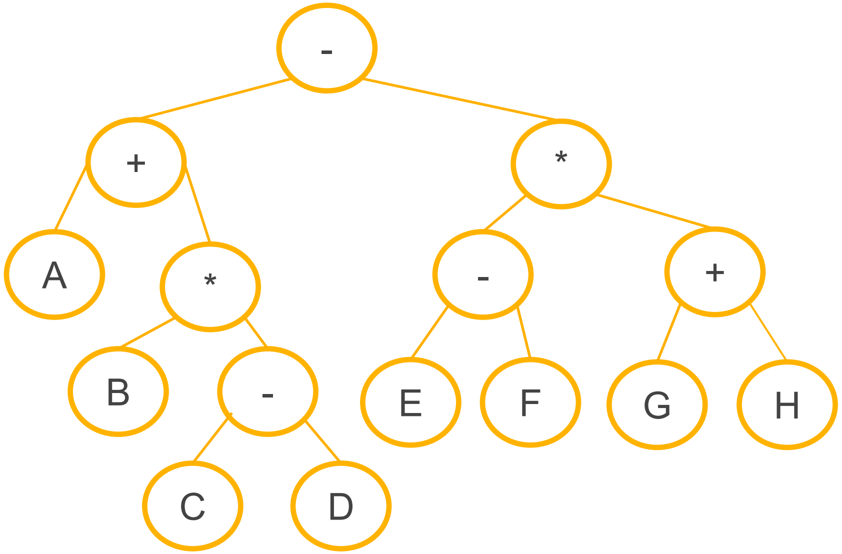
Binary Expression Tree

((A + B) – (C / D)) + (E * (F/ G))



Binary Expression Tree

(A + (B * (C – D))) – (E – F * (G + H))



Binary Tree Traversal

- to visit all the nodes of the tree in order
- not commonly used as finding, inserting, deleting nodes
- useful in some circumstances and simpler than deleting of nodes

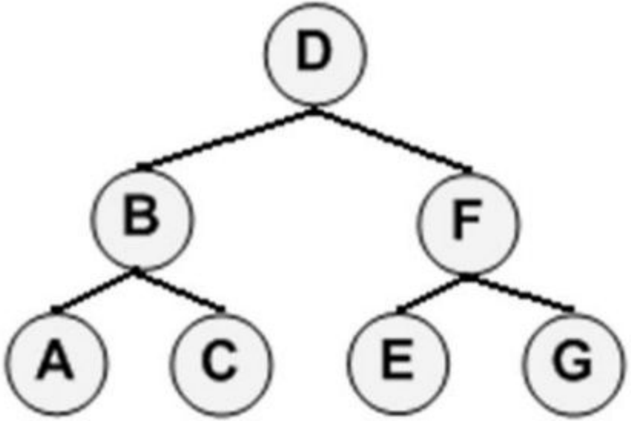
  - Inorder

  - Preorder

  - Postorder

Binary Tree Traversal : Inorder Traversal

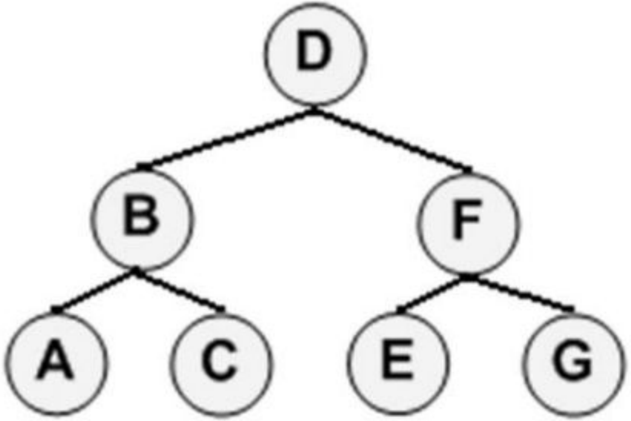
- If the node has a left subtree:
  - Traverse the left subtree in preorder (recursive call). Once completed, proceed to step 2.
  - Otherwise, proceed to step 2.
- Read the root node
- If the node has a right subtree:
  - Traverse the right subtree in preorder (recursive call).



Answer:
A B C D E F G

Binary Tree Traversal : Preorder Traversal

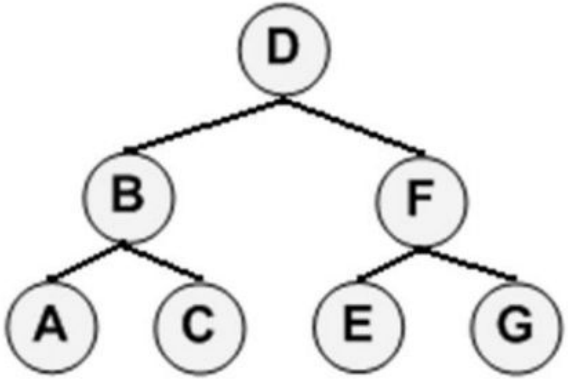
- Read the root node.
- If the node has a left subtree:
  - Traverse the left subtree in preorder (recursive call). Once completed, proceed to step 3.
  - Otherwise, proceed to step 3.Read the root node
- If the node has a right subtree:
  - Traverse the right subtree in preorder (recursive call).



Answer:
D B A C F E G

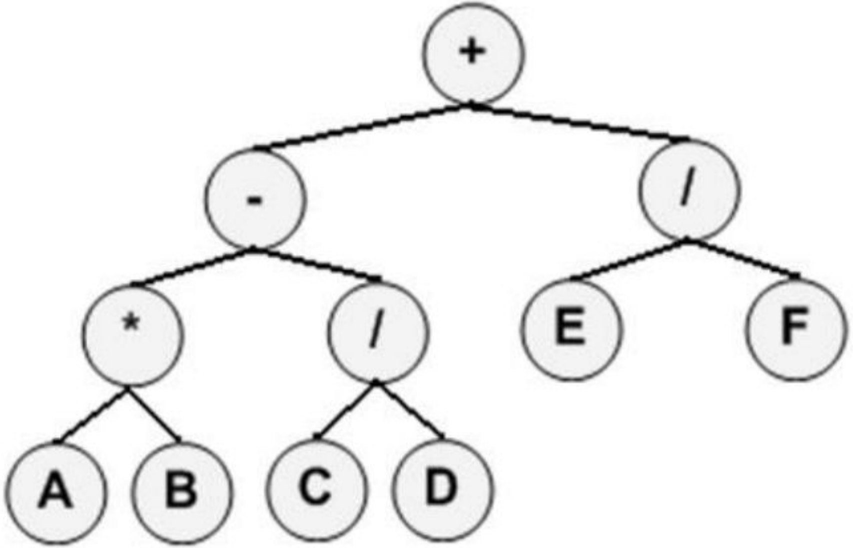
Binary Tree Traversal : Postorder Traversal

- If the node has a left subtree:
  - Traverse the left subtree in preorder (recursive call). Once completed, proceed to step 2.
  - Otherwise, proceed to step 2.
- If the node has a right subtree:
  - Traverse the right subtree in preorder (recursive call). Once completed, proceed to step 3.
  - Otherwise, proceed to step 3.
- Read the root node



Answer:
A C B E G F D

Binary Tree Traversal : Preorder Traversal



Inorder: A * B - C / D + E / F

Preorder: + - * A B / C D / E F

Postorder: A B * C D / - E F / +

Thank You 😊
Keep safe
and God bless!