



SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

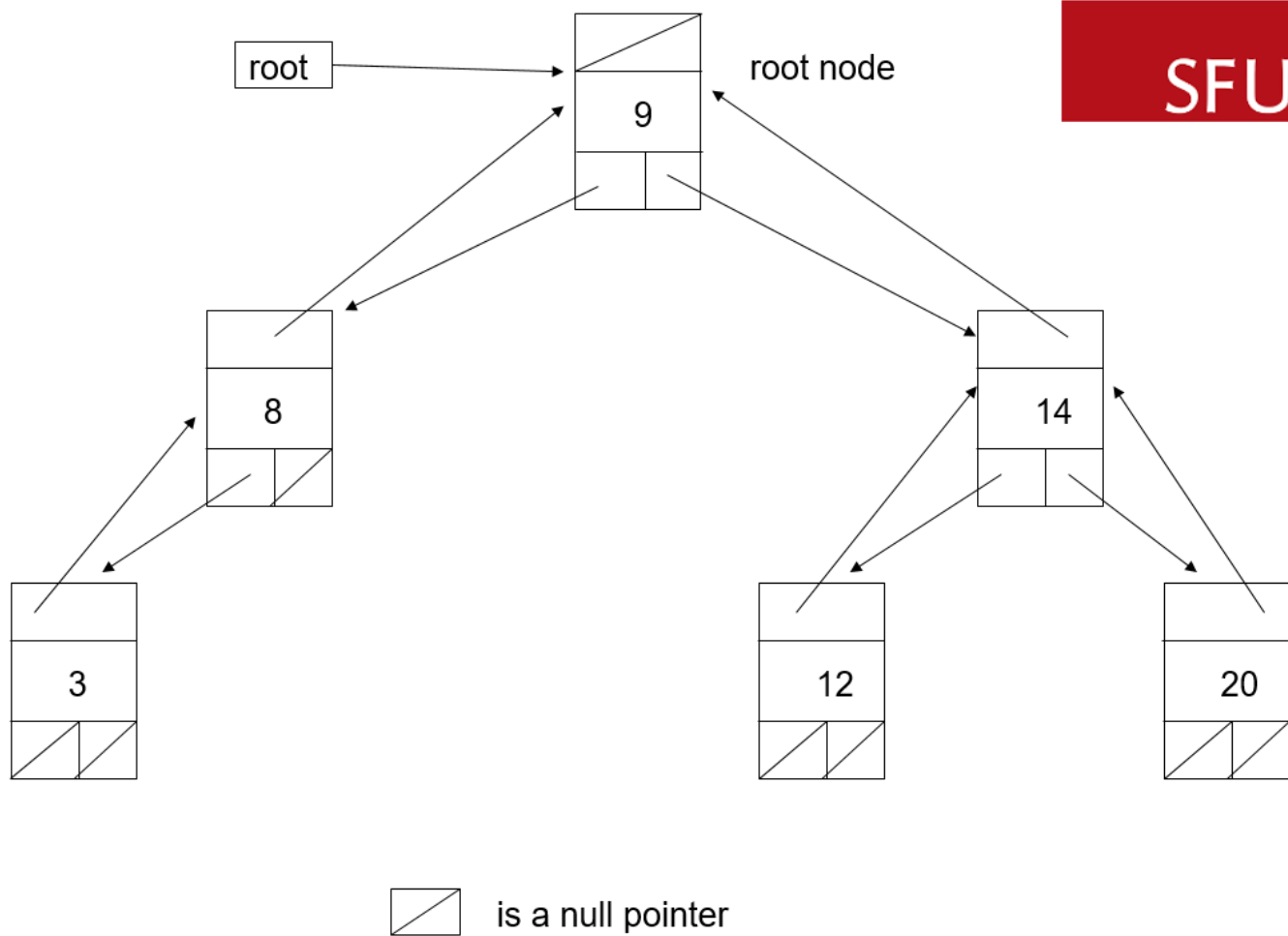
CMPT 125 - Introduction to Computing Science and Programming II

Binary Trees

Quick Recap – Binary Tree

Structure of a binary tree node

parent pointer	
data	
left child pointer	right child pointer



Quick Recap – Binary Tree

- Tree data structure where each vertex can have at most 2 children
- Vertices are either root, node or leaf
- At most 2^k nodes in level k
- Considered to be full if for all $k \leq \text{depth}$, it has 2^k nodes in level k
- For N vertices, the depth is at least $\log(N) - 1$ and at most $N-1$

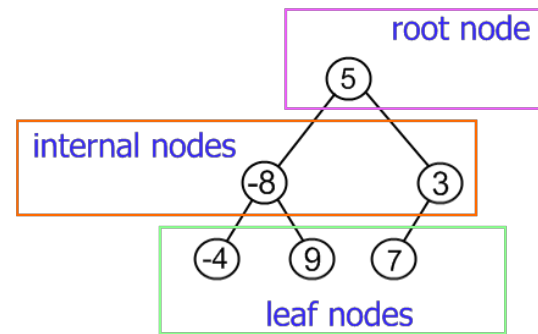
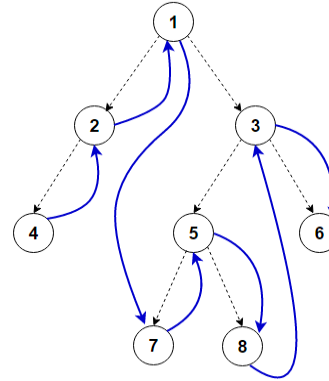


Fig1: Binary Tree

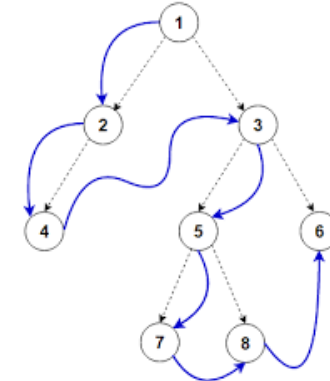
Quick Recap – Binary Tree Traversals

- InOrder Traversal:
 - Visit left subtree
 - Visit root node
 - Visit right subtree
- PreOrder Traversal:
 - Visit root node
 - Visit left subtree
 - Visit right subtree
- PostOrder Traversal:
 - Visit left subtree
 - Visit right subtree
 - Visit root node



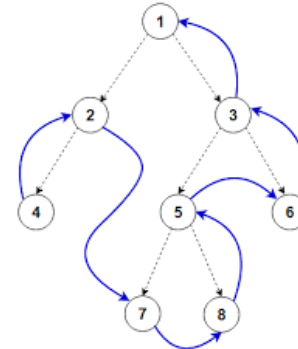
Inorder: 4, 2, 1, 7, 5, 8, 3, 6

Fig2: InOrder Traversal



Preorder: 1, 2, 4, 3, 5, 7, 8, 6

Fig3: PreOrder Traversal



Postorder: 4, 2, 7, 8, 5, 6, 3, 1

Fig4: PostOrder Traversal

Quick Recap – Binary Tree Operations

- `set_left_child(BTnode_t* parent, BTnode_t* left_child)`: set a node to be the left child of another node
- `set_right_child(BTnode_t* parent, BTnode_t* right_child)`: set a node to be the right child of another node
- `is_leaf(BTnode_t* root)`: check if a node is leaf or not
- `size(BTnode_t* root)`: returns the number of nodes in the tree
- `height(BTnode_t* root)`: returns the height/depth of the tree
- `print_pre_order(BTnode_t* root)`: traverses the tree in pre-order way and prints the nodes
- `print_in_order(BTnode_t* root)`: traverses the tree in in-order way and prints the nodes
- `print_post_order(BTnode_t* root)`: : traverses the tree in post-order way and prints the nodes

Exercise

- Read and understand the functions defined in BTnode.c
- Implement the functions:
 - `count_leaves(BTnode_t* root)`: Counts the number of leaves in the tree
 - `in_order_to_array(BTnode_t* root)`: Puts all elements of tree in an array with in-order traversal
 - `are_equal(BTnode_t* root1, BTnode_t* root2)`: Checks if two nodes of the tree are equal(value and children)
 - `BTnode_t* reconstruct_tree(int* inorder, int* preorder, int n)`: Reconstructs a tree given the number of nodes, the inorder traversal and the preorder traversal

Add more test cases to test the functions you implement

Steps to compile code

- Unzip and open the directory in VSCode
- `> make`
- `> ./test_BT`