# 

# Department of Computing

**CS 250: Data Structures and Algorithms**

**Lab 12: Binary Search Trees – Part B**

**Date: August 3, 2023**

# Instructor: Dr. Syed Imran Ali

# Lab : Implementation of Binary Search Tree – Part B

**Introduction**

This lab is based on the implementation of Binary Search tree and its functions.

**Objectives**

The objectives of this lab are the following:

* Become familiar with implementation of binary search trees
* Study some statistics of binary search trees
* Write simple applications using binary search tree

**Tools/Software Requirement**

Visual Studio 2012 or gcc or g++

**Description**

In computer science, a binary search tree (BST), which may sometimes also be called an ordered or sorted binary tree, is a node-based binary tree data structure which has the following properties:

* The left sub-tree of a node contains only nodes with keys less than the node's key.
* The right sub-tree of a node contains only nodes with keys greater than the node's key.
* Both the left and right sub-trees must also be binary search trees.
* There must be no duplicate nodes.

In this lab, you will expand implement binary search tree, study some statistical properties of BST and write a simple application using the BST.

Here is a template of how your class/structure looks like.

class BST\_Node{

Template data;

BST\_Node \*LeftChild;

BST\_Node \*RightChild;

};

**Lab Task**

You are required to upload the lab tasks on LMS and the name of that tasks must be in this format

FullName\_reg#\_task#.cpp

Remember to comment your code properly. Inappropriate or no comment will result in the deduction of marks.

**Tasks**

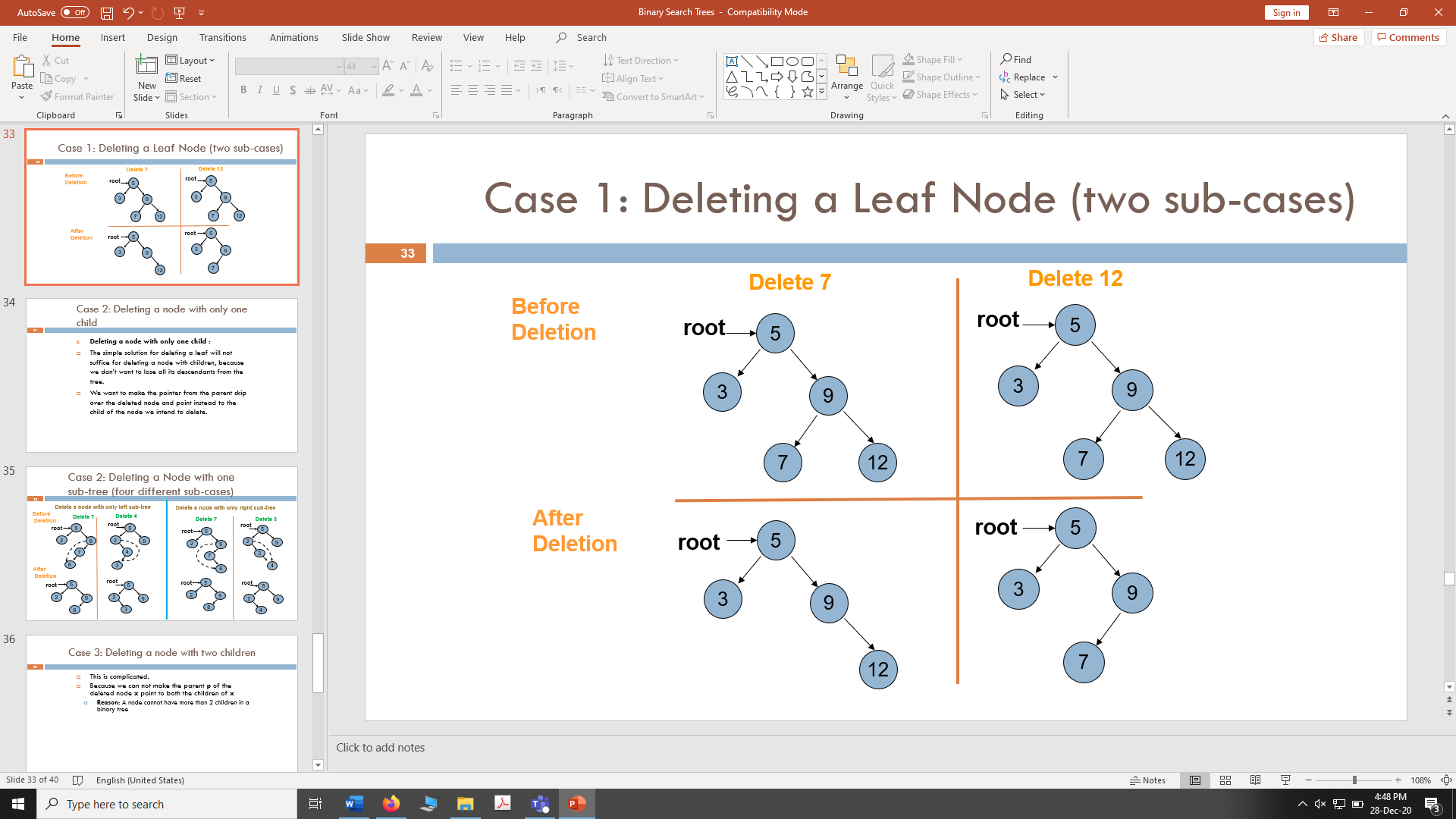
**Previous Tasks:** In the previous lab you implemented the following operation of a binary search tree:

* bool IsEmpty();
* void Search(template value)
* Void InsertWithoutDuplication(template value)
* Void InsertWithDuplication(template value)
* Traversing a binary tree in pre-order, in-order and post-order.
* Implement a function that prints the smallest value of a BST.
* Implement a function that prints the largest value of a BST.
* Implement a function to calculate the height of a BST.
* Implement a function that calculates the depth of a BST.

**In today’s lab**, your task is to implement the following operations of a binary search tree:

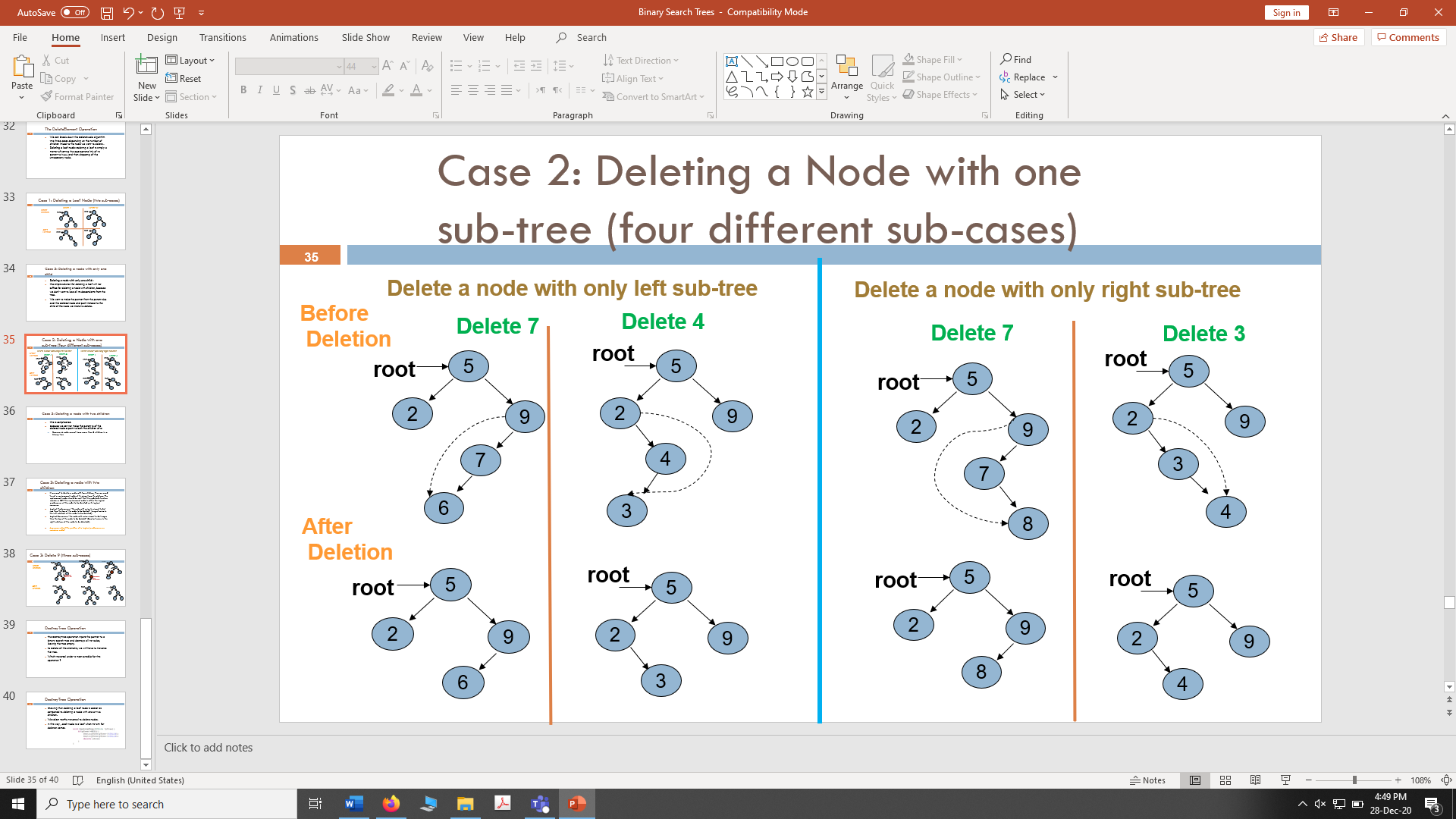
1. Deleting a node. You may call your search(value) function. You should implement all the three cases:
   1. **Deleting a leaf node.**

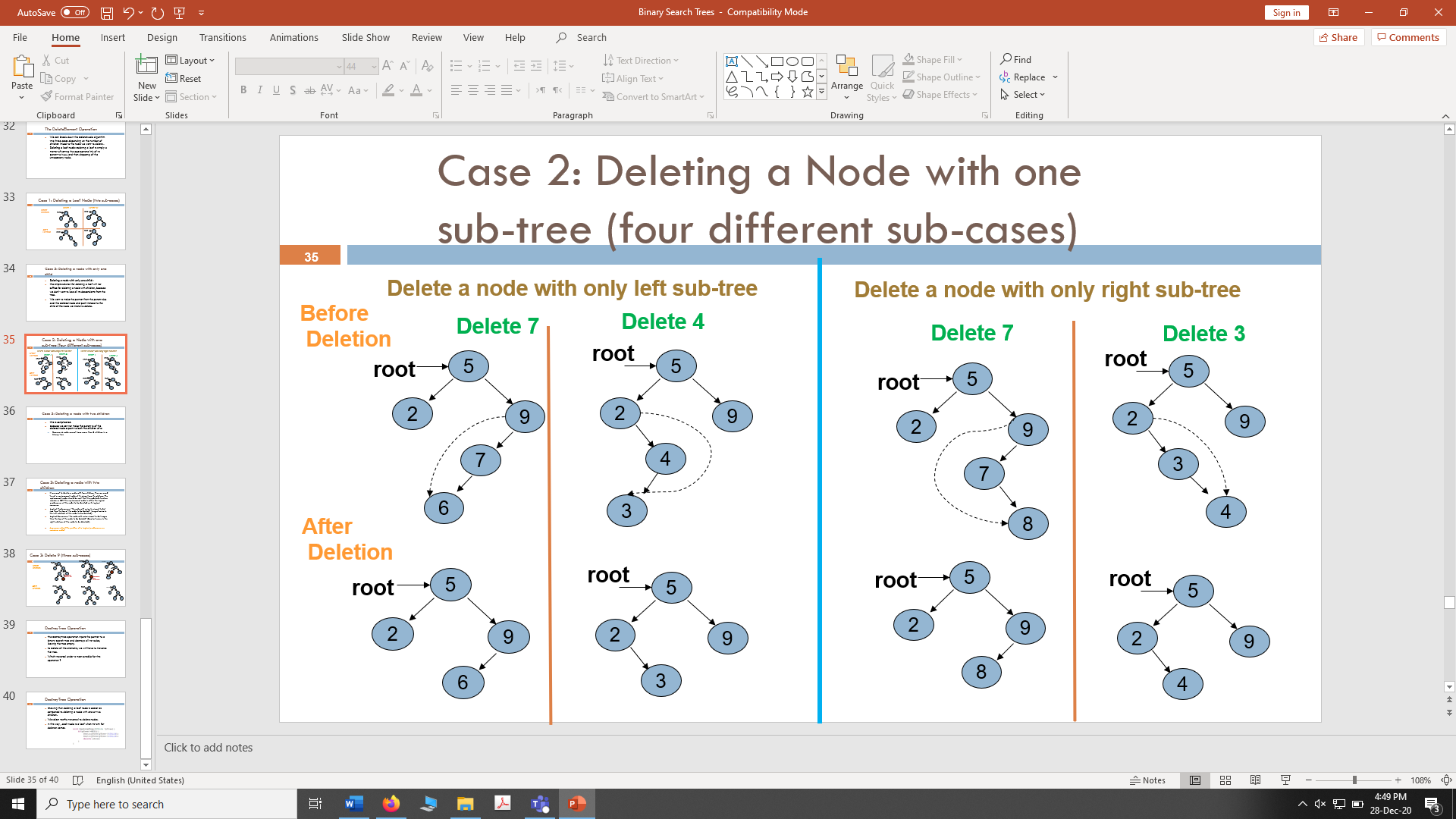
Include the case in which the node being deleted is a root node.



* 1. **Deleting a node with only one sub-tree.**

Include the case in which the node being deleted is a root node.



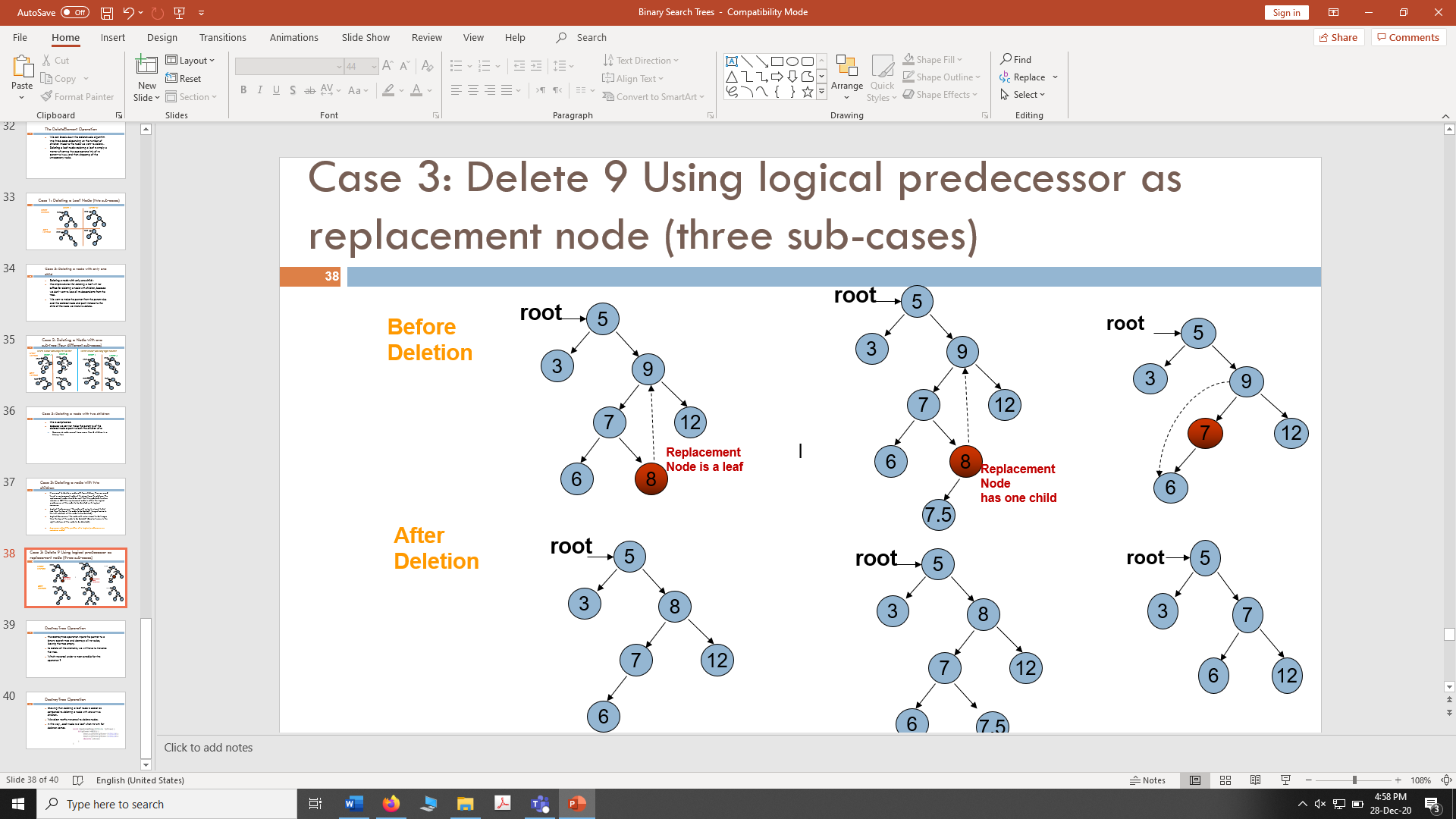


* 1. **Deleting a node with two children**: If we want to delete a node with two children, then we need to put a replacement node at its place from its sub-tree. The replacement node should be such that the updated structure remains a BST. The replacement node is either the logical predecessor of the node to be deleted or its logical successor.

**Logical Predecessor:** ”*the node with value to closest to but less than the key of the node to be deleted*” (largest value in the left sub-tree of the node to be deleted).

**Logical Successor:** ”*the node with value closest to but larger than the key of the node to be deleted*” (Smallest value in the right sub-tree of the node to be deleted).

In the below example, the node containing value 9 is deleted using its logical predecessor as replacement node. Include the case in which the node being deleted is a root node.



1. **Destroy tree:** Implement a function that destroys all nodes of a binary tree leaving the tree empty.
2. Implement a function that traverses a BST and **prints** only its **leaf nodes.**
3. Implement a function that traverses all nodes of a BST once, and counts the number of leaves, nodes with only left child, nodes with only right child and nodes with two children in it. You should maintain a separate counter variable for each of the four types of nodes.
4. Implement a function that **deletes** all **leaf** nodes of a **given BST only.**
5. Implement a function that **deletes** only those nodes from a **given** **BST** that have only **left sub-tree.** Only those nodes should be deleted that have left branch in the original tree.

**Deliverable**

Students are required to upload the lab task on LMS before the deadline.