# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

**Class: BS****CS & BSAI**

# Lab 5: Implementation of Binary Search Tree

|  |  |
| --- | --- |
| Student Name |  |
| CMS Id |  |

**Date: 7th October, 2025**

**Time: 02 pm – 05 pm**

**Course Instructor: Dr. Aimal Tariq**

**Lab Engineer: Ms Areeba Rameen**

# 

# Lab 5: Implementation of Binary Search Tree

**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**

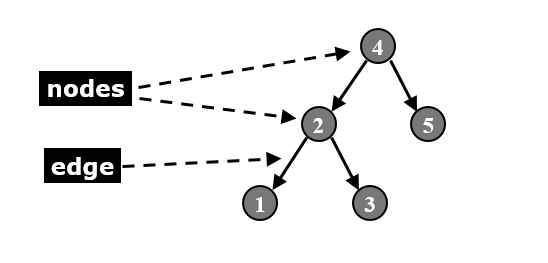
Visual Studio 2012 or gcc or g++

**Helping Material**

Lecture slides, text book

**Description**

* Trees are a hierarchical data structure of nodes
* Each position in the tree is called a node
* Nodes are linked by edges



**Tree Terminology (Parent / Child)**

* **Root:** node without parent
  + A parent node references one or more nodes (children nodes) that are “lower” in the tree hierarchy
  + Except for the root (no parent), every node has exactly one parent (by definition)
  + A tree has only one root node

**Leaf:** A node is a leaf if it has no children

**Types of Trees**

|  |  |
| --- | --- |
| **General tree –** a node can have any number of children | **Binary tree –** a node can have at most two children |
|  |  |

**Binary Search Tree**

Binary tree

* The simplest form of tree is a Binary Tree
* A Binary Tree consists of
  + (a) A **node** (called the **root node**) and
  + (b) **Left** and **right subtrees**
  + Both the subtrees are themselves binary trees
    - Note: this is a recursive definition
* (A node can’t have more than 2 children)

**Full binary tree:** All leaves on the same level and every node has either zero or two children.

**Complete binary tree:** Leaves are filled from left to right on one level before moving to next level.

****

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:

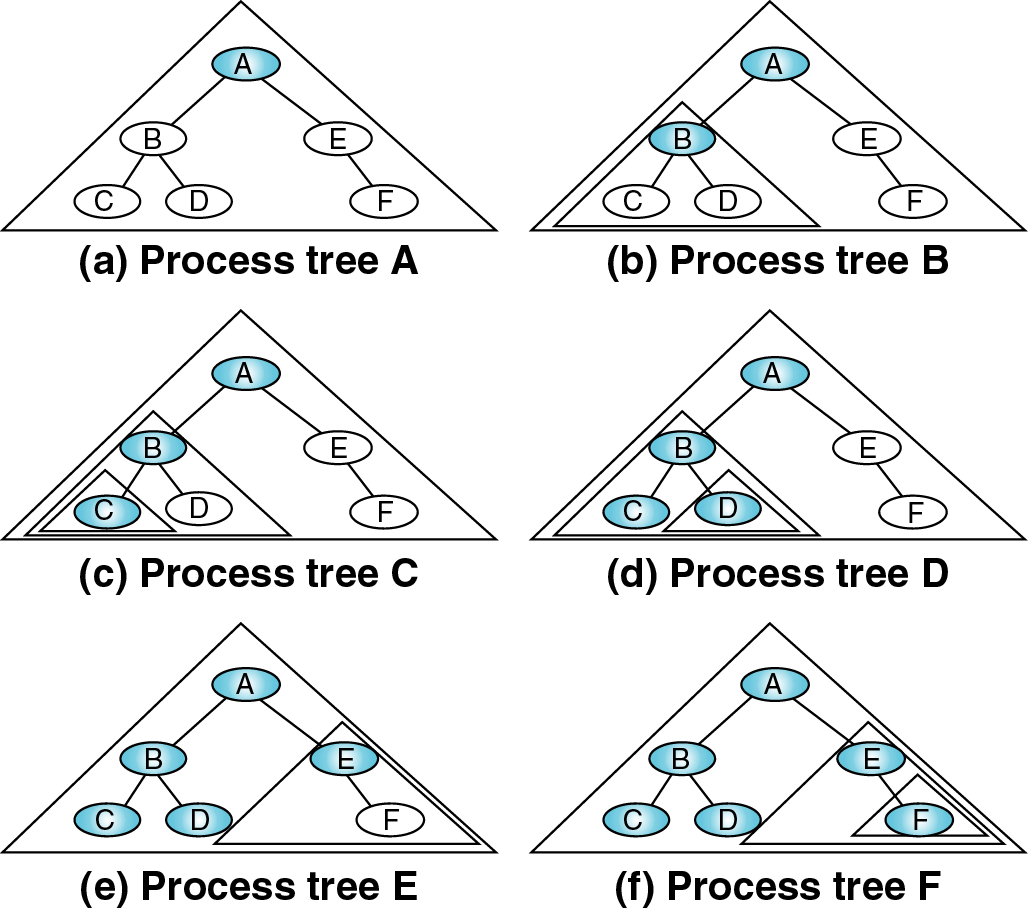
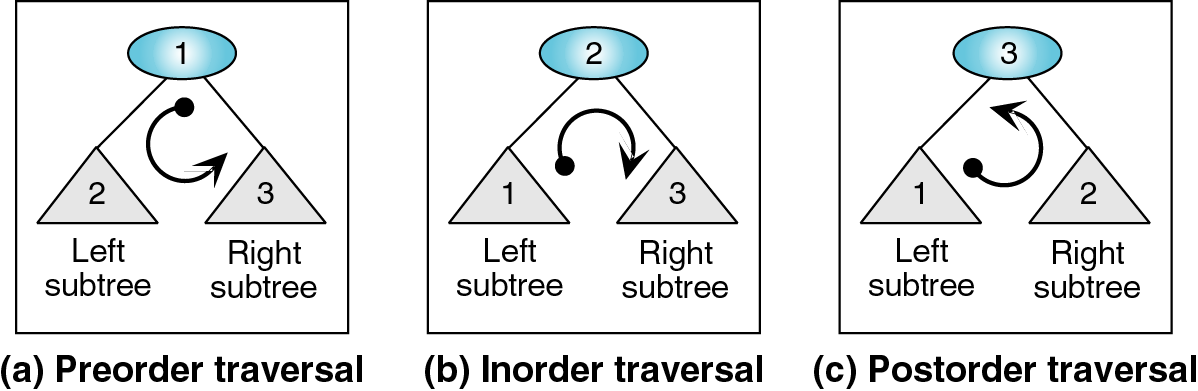
* The left sub-tree of a node contains only nodes with keys less than the node's key.
* The right sub-tree of a node contains only nodes with keys greater than the node's key.
* Both the left and right sub-trees must also be binary search trees.
* There must be no duplicate nodes. (No two entries in a binary search tree can have equal keys.)

|  |  |
| --- | --- |
| Here is a template of how your class/structure looks like.  struct tree\_node{  tree\_node\* left;  tree\_node\* right;  int data;  }; |  |

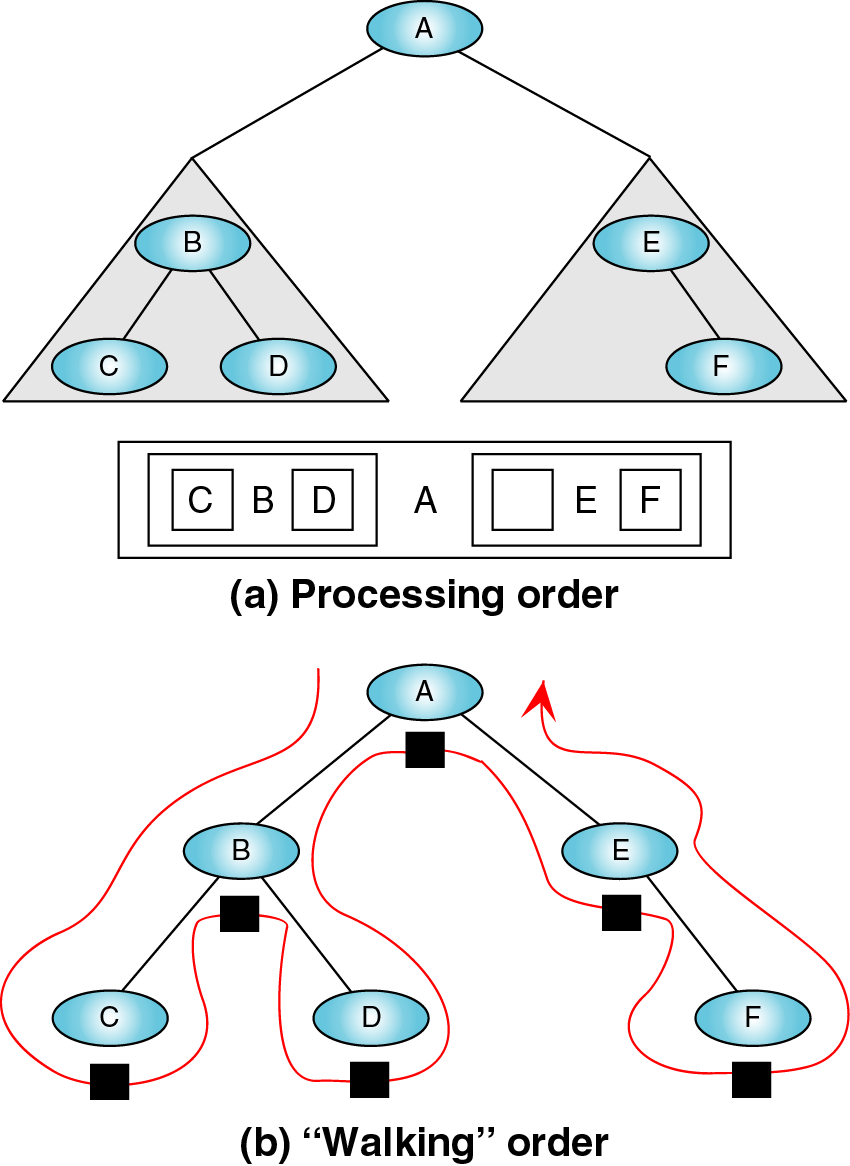
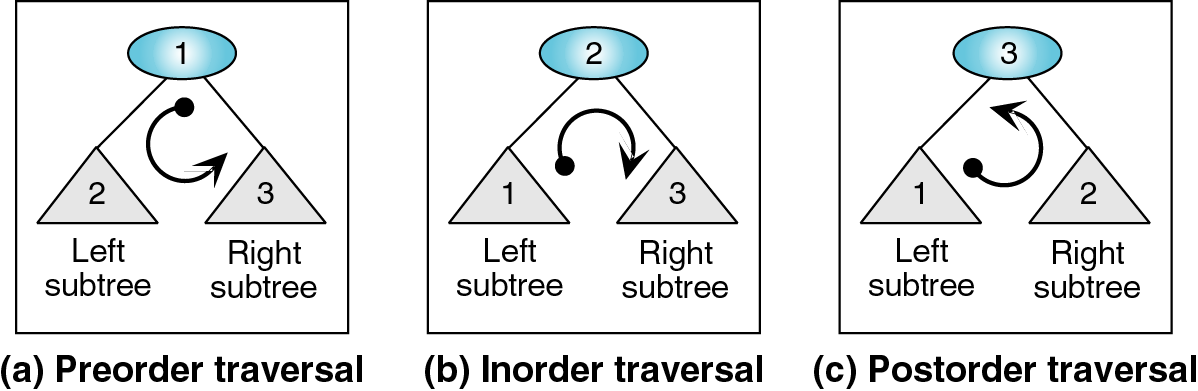
|  |  |
| --- | --- |
| **Common Operations**   * Tree traversal * Node addition * Node deletion * Destroy | **Traversal of Binary Trees**   * Pass through all nodes of tree * Inorder traversal * Preorder traversal * Postorder traversal |



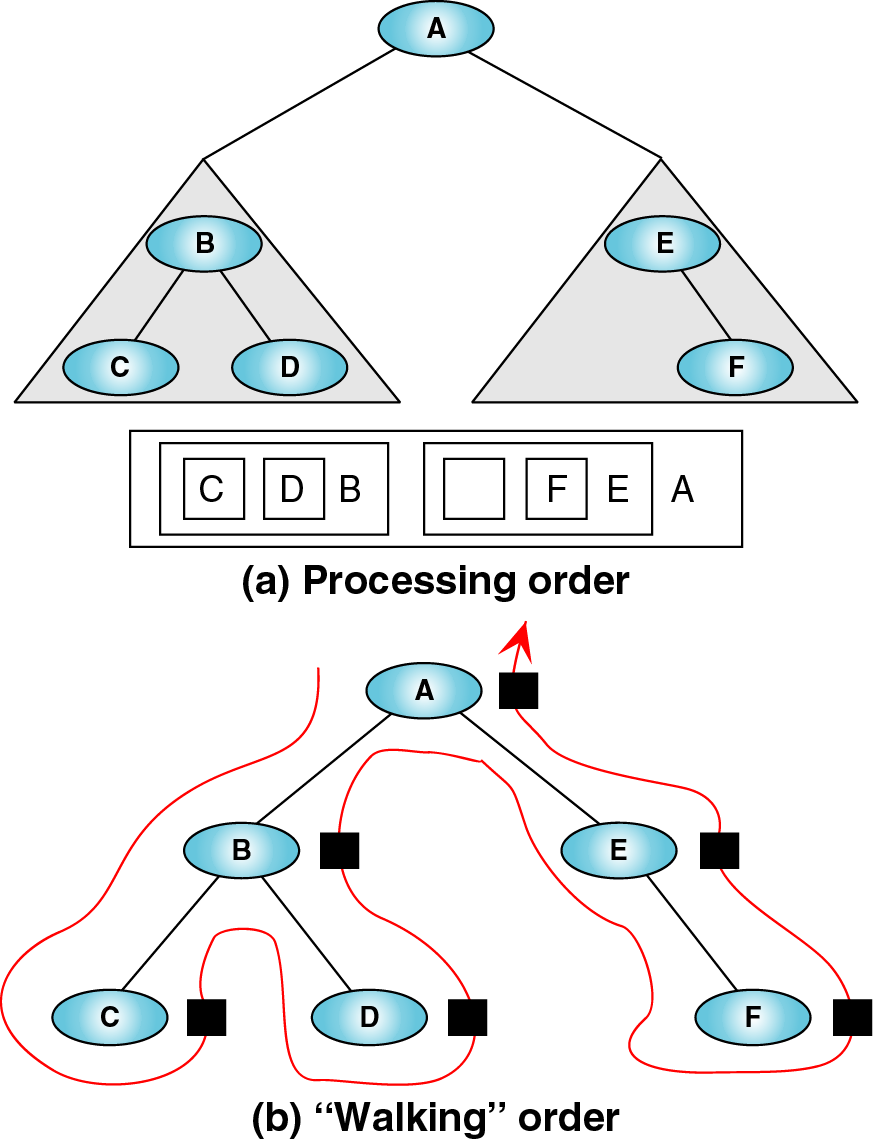
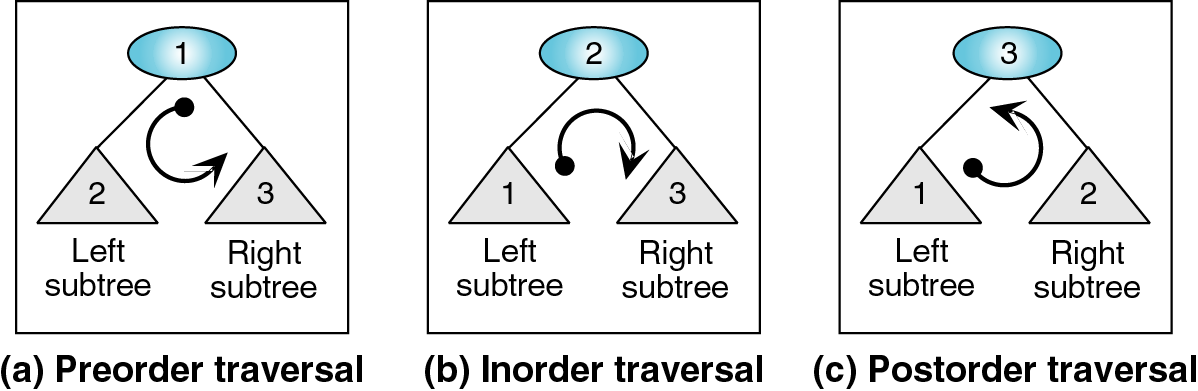
**Preorder Traversal**

** **

**Inorder Traversal**

****  ****

**Postorder Traversal**

** **

**Visiting and Traversing a Node**

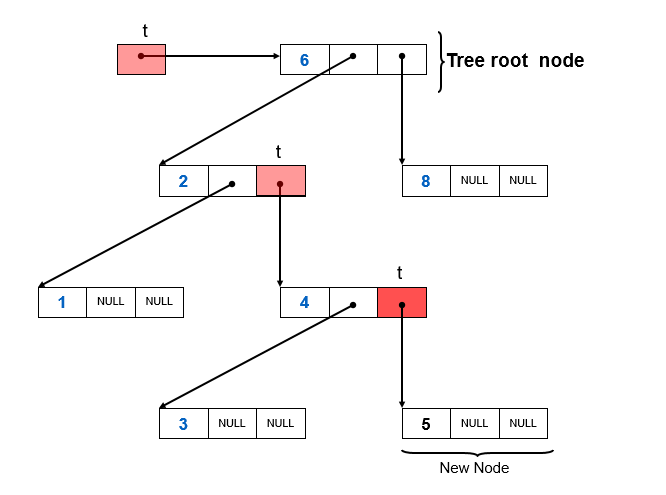
* Many applications require that all of the nodes of a tree be “visited”.
* Visiting a node may mean printing contents, retrieving information, making a calculation, etc.
* Traverse: To visit all the nodes in a tree in a systematic fashion.
  + A traversal can pass through a node without visiting it at that moment.

**Inserting a value into a BST:**

* The first value inserted goes at the root
* Every node inserted becomes a leaf
* Descend left or right depending on value



**Inserting Item 5 to the Tree**



**Searching the tree**

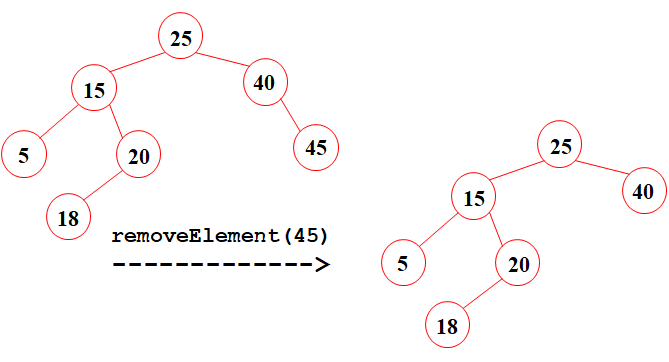
* Start at the root
* Is target = value at current node?
  + We’re done
* Is target < value at current node?
  + Yes: search left subtree
  + No: search right subtree

**Deleting a value into a BST:**

* There are **three possible cases** when deleting a node in a BST.
* The BST property (left < root < right) must always remain valid.
* Depending on the node’s children, different strategies are used.

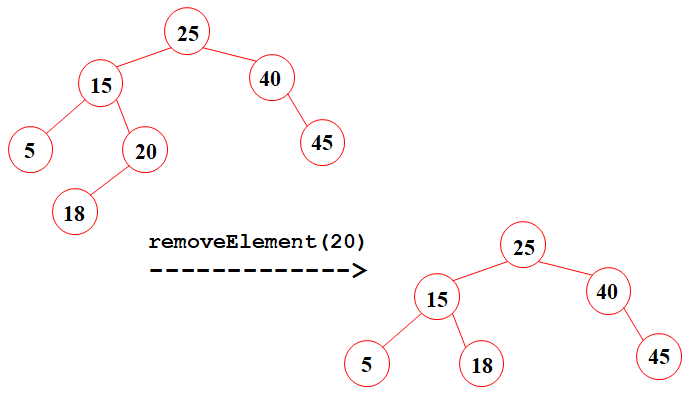
**Case 1: Node to be deleted is a Leaf Node (No Children)**

* Simply remove the node.
* No other links are affected.



**Case 2: Node to be deleted has One Child**

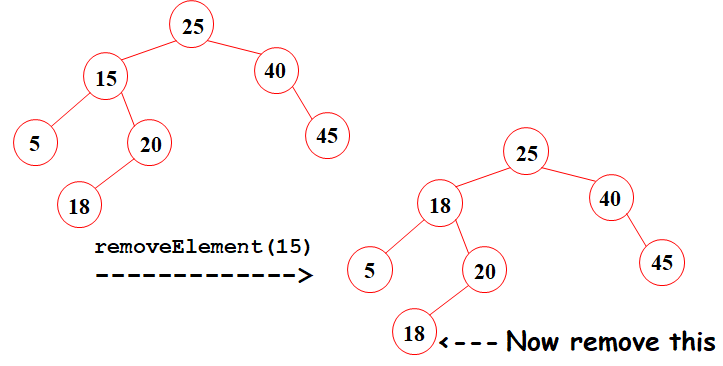
• Bypass the node by linking its parent directly to its child.  
• The deleted node is removed, and its child takes its place.



**Case 3: Node to be deleted has Two Children**

•Find the inorder successor (smallest node in the right subtree) or inorder predecessor (largest in the left subtree).

**x occurs at a node with two children**: first replace **x** with smallest value in right subtree of **x**. This value occurs at a node with no left child. So we can delete this node using one of the two previous cases.



**Tasks**

Implement the following operations of Binary Search Tree ADT.

Write a main method which contains a menu to take user’s inputs. The menu will allow the user to do the following operations:

* 1. Insert new data
  2. In-Order Traversal
  3. Pre-Order Traversal
  4. Post-Order Traversal
  5. Delete an item

1. For insertion operation, the user will give a value which will be added in the binary search tree at its correct position.
2. For in-order, pre-order, and post-order traversals, the user should see the traversed tree as output in correct order.
3. On removal of an item, the nodes should connect correctly so that it does not break the data hierarchy within the tree.
4. Test your program with different inputs from smaller array e.g. of size 10 to larger arrays of size 100.

**Important Note:** Practice your knowledge of OOP with C++ when creating a solution. Remember to comment your code properly. Inappropriate or no comment may result in deduction of marks.

**Solution**

|  |
| --- |
| Solution |
| Task 1 code Screen Shots (execution: Insertion, 3 types of traversals, deletion):  Task 1 output screenshot: |

### Deliverables

Compile a single word document by filling in the solution part and submit this Word file on LMS. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS.