function:
Preorder (Psuedocode) (N-L-R)

Print
Void Preorder(Treenode *r)
1) If r == null -> Return (Stop)
2) Print r->Value • Access value + Print
3) Print Preorder(r-> left) • Recursive (access val + Print)
4) Print Preorder(r-> right) • Recursive (access val + Print)

Function:
In-order (Psuedocode) (L-N-R)

Print
Void Inorder(Treenode *r)
1) If r == null -> return (Stop)
2) In-order(r-> Left) • Recursive (until null, then Print)
3) Print r->value • Access val + Print
4) In-order(r-> right) • Recursive

Function:
Post-order (Psuedocode) (L R N)

Void postorder(TreeNode *r)
1) If r == null -> return (Stop)
2) Postorder(r-> left) recursive (Till left = null)
3) Postorder(r -> right) recursive (Till right = null)
4) print r->value

Delete Pseudocode

1) IF r == null
   ↳ return r
2) Else if v < r->value // if val smaller go left
   ↳ r->left = deleteNode(r->left, v)
3) Else if v > r->value // if val larger go right
   ↳ r->right = deleteNode(r->right, v)
4) Else // if value matches
   ↳ if r->left == null // node w/ only right child
      ↳ temp = r->right
      ↳ delete r
      ↳ return temp
   ↳ Else if r->right == null // node w/ only left child
      ↳ temp = r->left
      ↳ delete r
      ↳ return temp
   ↳ Else
      ↳ temp = min right val node (r->right)
      ↳ r->value = temp->value
        r->right = deleteNode(r->right, temp->value)

return r

Tests

Insert_Node:
① Insert duplicate values; Should prompt user to reenter
② Insert Node->value >< root->value; Should create a left and right subtree.

Search_Node:
① Search for node->value that doesn't exist; Should prompt that val does not exist

delete_node:
① Delete root node->value; root->value should be replaced by min val on right subtree.

Pre_order_Print:
↳ 77, 56, 45, 57, 90, 89, 98

In-order_Print:
↳ 45, 56, 57, 77, 89, 90, 98

Post-order_Print:
45, 57, 56, 89, 90, 98