Flatten binary tree to Linked List

You are given a binary tree consisting of **'n'** nodes.

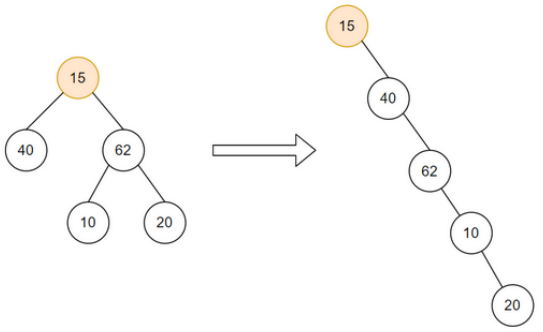
Convert the given binary tree into a linked list where the linked list nodes follow the same order as the pre-order traversal of the given binary tree.

Use the right pointer of the binary tree as the "next" pointer for the linked list and set the left pointer to NULL.

Use these nodes only. Do not create extra nodes.

Example :

Input: Let the binary be as shown in the figure:

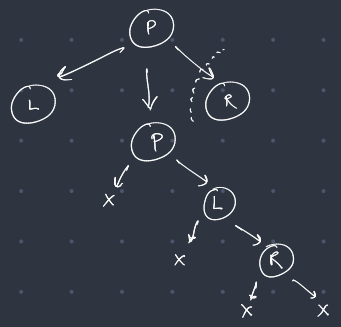


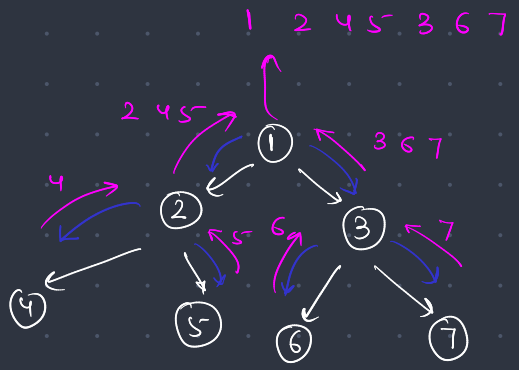
Output: Linked List: 15 -> 40 -> 62 -> 10 -> 20 -> NULL

Explanation: As shown in the figure, the right child of every node points to the next node, while the left node points to null.

Also, the nodes are in the same order as the pre-order traversal of the binary tree.

→ Approach : Using recursion





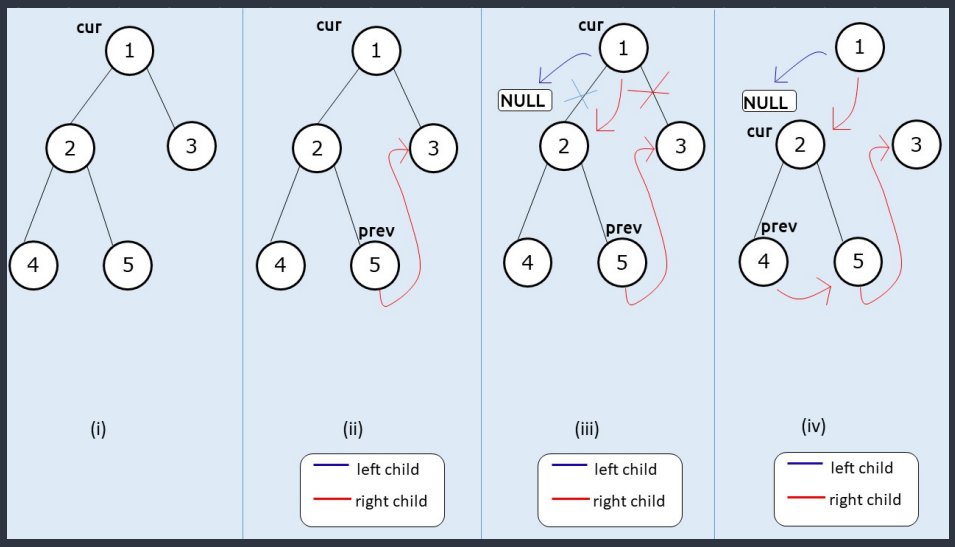

```
TreeNode<int>* solve(TreeNode<int>* root) {
    if(!root->left && !root->right) {
        return root;
    }
    TreeNode<int>* left = NULL;
    TreeNode<int>* right = NULL;
    if(root->left) left = solve(root->left);
    if(root->right) right = solve(root->right);

    root->left = NULL;
    root->right = left;
    TreeNode<int>* curr = root;
    while(curr->right != NULL) curr = curr->right;
    if(right) curr->right = right;
    return root;
}

void flattenBinaryTree(TreeNode<int>* root)
{
    root = solve(root);
}
```

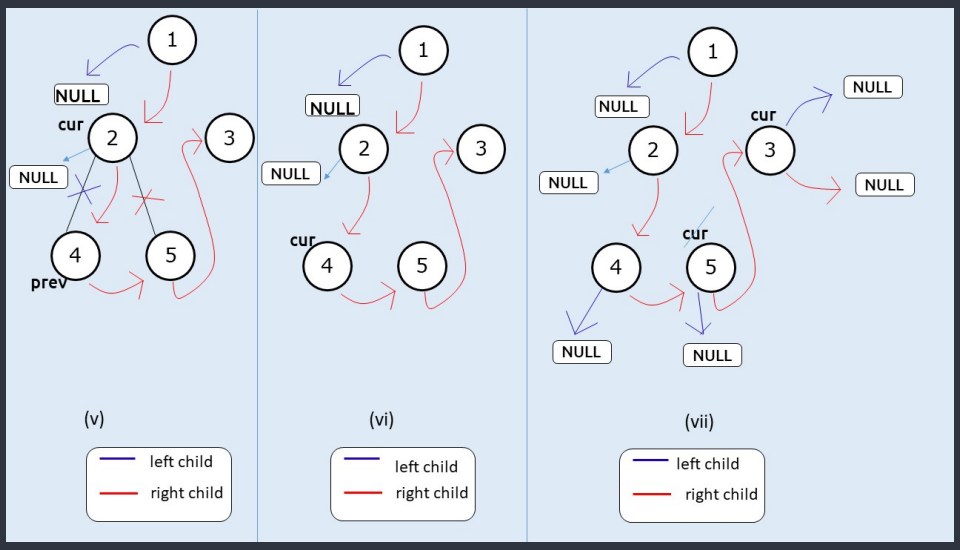
Time Complexity : O(N)

Space Complexity : O(H)

→ Using Stack

→ Using Morris traversal:-






```
void flattenBinaryTree(TreeNode<int> *root) {
  TreeNode<int> *cur = root;
  TreeNode<int>* prev = NULL;
  while (cur) {
    if (cur->left) {
      TreeNode<int> *pre = cur->left;
      while (pre->right) {
        pre = pre->right;
      }
      pre->right = cur->right;
      cur->right = cur->left;
      cur->left = NULL;
    }
    cur = cur->right;
  }
}
```

Time Complexity : O(n)

Space Complexity : O(1)

→ Using Extra Space:-

```
void traverse(TreeNode<int> *root, vector<TreeNode<int> *> &v) {
  if (root == NULL) {
    return;
  }
  v.push_back(root);
  if (root->left)
    traverse(root->left, v);
  if (root->right)
    traverse(root->right, v);
}

void flattenBinaryTree(TreeNode<int> *root) {
  if (root == NULL)
    return;
  vector<TreeNode<int> *> v;
  traverse(root, v);
  root = NULL;
  TreeNode<int> *temp = NULL;
  for (int i = 0; i < v.size(); i++) {
    if (root == NULL) {
      root = v[i];
      root->left = NULL;
      temp = v[i];
    } else {
      temp->right = v[i];
      temp->left = NULL;
      temp = temp->right;
    }
  }
}
```

Time Complexity : O(N)

Space Complexity : O(N)