

109.Convert Sorted List to Binary Search Tree

```
#include <stdio.h>

#include <stdlib.h>

/* Definition for singly-linked list */
struct ListNode {
    int val;
    struct ListNode *next;
};

/* Definition for binary tree node */
struct TreeNode {
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
};

/* Create a new list node */
struct ListNode* createListNode(int val) {
    struct ListNode* node = (struct ListNode*)malloc(sizeof(struct ListNode));
    node->val = val;
    node->next = NULL;
    return node;
}

/* Create a new tree node */
struct TreeNode* createTreeNode(int val) {
    struct TreeNode* node = (struct TreeNode*)malloc(sizeof(struct TreeNode));
    node->val = val;
    node->left = node->right = NULL;
    return node;
}
```

```
}
```

```
/* Function to display linked list */
```

```
void displayList(struct ListNode* head) {
```

```
    struct ListNode* temp = head;
```

```
    while (temp != NULL) {
```

```
        printf("%d-> ", temp->val);
```

```
        temp = temp->next;
```

```
    }
```

```
    printf("NULL\n");
```

```
}
```

```
/* Function to create sorted linked list from user input */
```

```
struct ListNode* createSortedList(int n) {
```

```
    int val;
```

```
    struct ListNode *head = NULL, *tail = NULL;
```

```
    for (int i = 0; i < n; i++) {
```

```
        printf("Enter element %d: ", i + 1);
```

```
        scanf("%d", &val);
```

```
        struct ListNode* node = createListNode(val);
```

```
        if (head == NULL) {
```

```
            head = tail = node;
```

```
        } else {
```

```
            tail->next = node;
```

```
            tail = node;
```

```
        }
```

```
    }
```

```
    return head;
```

```
}
```

```
/* Convert sorted linked list to height-balanced BST */
```

```

struct TreeNode* sortedListToBST(struct ListNode* head) {
    if (head == NULL) return NULL;

    if (head->next == NULL) {
        struct TreeNode* root = createTreeNode(head->val);
        return root;
    }

    /* Find middle using slow-fast pointer */
    struct ListNode *slow = head, *fast = head, *prev = NULL;

    while (fast != NULL && fast->next != NULL) {
        prev = slow;
        slow = slow->next;
        fast = fast->next->next;
    }

    /* Disconnect left half */
    prev->next = NULL;

    /* Middle element becomes root */
    struct TreeNode* root = createTreeNode(slow->val);

    /* Recursively build left and right subtrees */
    root->left = sortedListToBST(head);
    root->right = sortedListToBST(slow->next);

    return root;
}

/* Tree traversal functions */

```

```
void inorder(struct TreeNode* root) {  
    if (root == NULL) return;  
    inorder(root->left);  
    printf("%d ", root->val);  
    inorder(root->right);  
}
```

```
void preorder(struct TreeNode* root) {  
    if (root == NULL) return;  
    printf("%d ", root->val);  
    preorder(root->left);  
    preorder(root->right);  
}
```

```
void postorder(struct TreeNode* root) {  
    if (root == NULL) return;  
    postorder(root->left);  
    postorder(root->right);  
    printf("%d ", root->val);  
}
```

```
int main() {  
    int n;  
    printf("Enter number of elements in sorted linked list: ");  
    scanf("%d", &n);  
  
    printf("Enter sorted elements:\n");  
    struct ListNode* head = createSortedList(n);  
  
    printf("Linked list:\n");  
    displayList(head);  
}
```

```

struct TreeNode* bstRoot = sortedListToBST(head);

printf("\nBST In-order Traversal: ");
inorder(bstRoot);

printf("\nBST Pre-order Traversal: ");
preorder(bstRoot);

printf("\nBST Post-order Traversal: ");
postorder(bstRoot);

printf("\n");

return 0;
}

```

OUTPUT:

```

Microsoft Windows [Version 10.0.26200.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\DS-Lab\Leetcode_Problems>cd "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\DS-Lab\Leetcode_Problems\" && gcc 109_ConvertListToBST.c -o 109_ConvertListToBST
Enter number of elements in sorted linked list: 5
Enter sorted elements:
Enter element 1: 1
Enter element 2: 2
Enter element 3: 3
Enter element 4: 4
Enter element 5: 5
Linked list:
1 -> 2 -> 3 -> 4 -> 5 -> NULL

BST In-order Traversal: 1 2 3 4 5
BST Pre-order Traversal: 3 2 1 5 4
BST Post-order Traversal: 1 2 4 5 3

C:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\DS-Lab\Leetcode_Problems>cd "c:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\DS-Lab\Leetcode_Problems\" && gcc 109_ConvertListToBST.c -o 109_ConvertListToBST
Enter number of elements in sorted linked list: 7
Enter sorted elements:
Enter element 1: -10
Enter element 2: -3
Enter element 3: 0
Enter element 4: 5
Enter element 5: 9
Enter element 6: 10
Enter element 7: 3
Linked list:
-10 -> -3 -> 0 -> 5 -> 9 -> 10 -> 3 -> NULL

BST In-order Traversal: -10 -3 0 5 9 10 3
BST Pre-order Traversal: 5 -3 -10 0 10 9 3
BST Post-order Traversal: -10 0 -3 9 3 10 5

C:\Users\Mohammed Javeed\OneDrive\Desktop\Javeed\DS-Lab\Leetcode_Problems>

```