**Write a program  
a. To construct a binary Search tree.  
b. To traverse the tree using all the methods i.e., in-order, preorder and postorder  
c. To display the elements in the tree.**

#include <stdio.h>

#include <stdlib.h>

struct TreeNode {

int data;

struct TreeNode\* left;

struct TreeNode\* right;

};

struct TreeNode\* createNode(int data) {

struct TreeNode\* newNode = (struct TreeNode\*)malloc(sizeof(struct TreeNode));

newNode->data = data;

newNode->left = newNode->right = NULL;

return newNode;

}

struct TreeNode\* insertNode(struct TreeNode\* root, int data) {

if (root == NULL) {

return createNode(data);

}

if (data < root->data) {

root->left = insertNode(root->left, data);

} else if (data > root->data) {

root->right = insertNode(root->right, data);

}

return root;

}

void inOrderTraversal(struct TreeNode\* root) {

if (root != NULL) {

inOrderTraversal(root->left);

printf("%d ", root->data);

inOrderTraversal(root->right);

}

}

void preOrderTraversal(struct TreeNode\* root) {

if (root != NULL) {

printf("%d ", root->data);

preOrderTraversal(root->left);

preOrderTraversal(root->right);

}

}

void postOrderTraversal(struct TreeNode\* root) {

if (root != NULL) {

postOrderTraversal(root->left);

postOrderTraversal(root->right);

printf("%d ", root->data);

}

}

void displayTree(struct TreeNode\* root) {

printf("In-order traversal: ");

inOrderTraversal(root);

printf("\n");

printf("Pre-order traversal: ");

preOrderTraversal(root);

printf("\n");

printf("Post-order traversal: ");

postOrderTraversal(root);

printf("\n");

}

int main() {

struct TreeNode\* root = NULL;

int choice, data;

do {

printf("1. Insert a node\n");

printf("2. Display tree\n");

printf("3. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data to insert: ");

scanf("%d", &data);

root = insertNode(root, data);

break;

case 2:

if (root == NULL) {

printf("Tree is empty.\n");

} else {

displayTree(root);

}

break;

case 3:

printf("Exiting program.\n");

break;

default:

printf("Invalid choice. Please try again.\n");

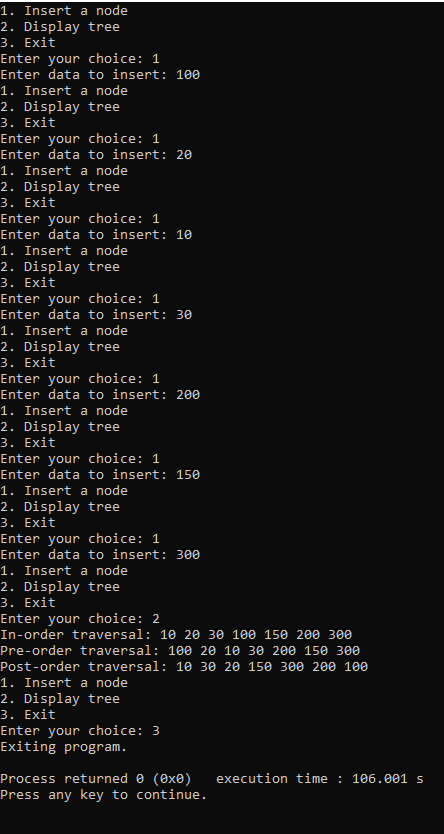
}

} while (choice != 3);

return 0;

}

**Output**

****

**Leet code**

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* struct ListNode \*next;

\* };

\*/

struct ListNode\* rotateRight(struct ListNode\* head, int k) {

if (head == NULL || k == 0) {

return head;

}

struct ListNode\* current = head;

int length = 1;

while (current->next != NULL) {

current = current->next;

length++;

}

k = k % length;

if (k == 0) {

return head;

}

current = head;

for (int i = 1; i < length - k; i++) {

current = current->next;

}

struct ListNode\* newHead = current->next;

current->next = NULL;

current = newHead;

while (current->next != NULL) {

current = current->next;

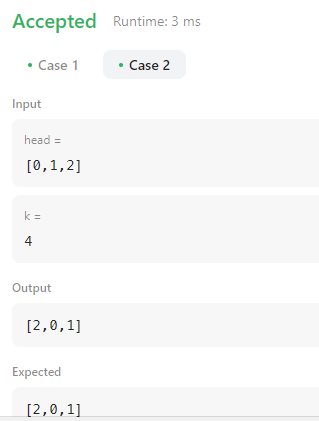
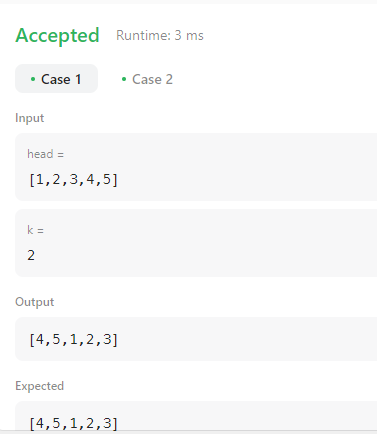
}

current->next = head;

return newHead;

}

**Output**

****