**Q. Count Total Nodes in Binary Search Tree**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* insert(struct Node\* root, int data) {

if (root == NULL) {

root = (struct Node\*)malloc(sizeof(struct Node));

root->data = data;

root->left = root->right = NULL;

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

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

} else {

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

}

return root;

}

int countTotalNodes(struct Node\* root) {

if (root == NULL) return 0;

return 1 + countTotalNodes(root->left) + countTotalNodes(root->right);

}

int main() {

struct Node\* root = NULL;

int choice, value;

do {

printf("\n1. Insert Node\n2. Count Total Nodes\n3. Exit\nEnter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

root = insert(root, value);

break;

case 2:

printf("Total Nodes: %d\n", countTotalNodes(root));

break;

case 3:

printf("Exiting...\n");

break;

default:

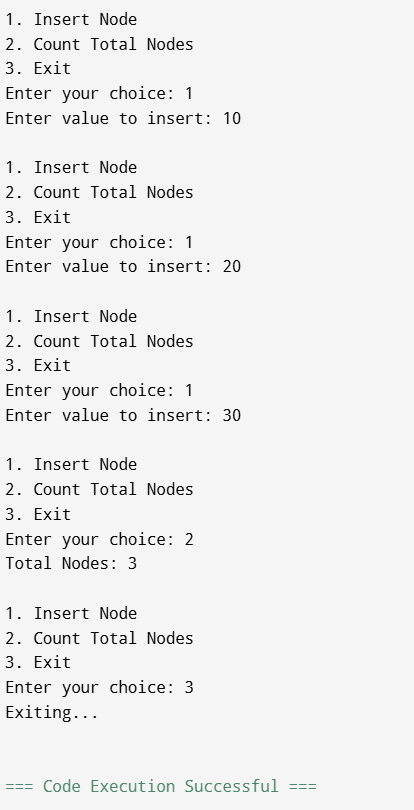
printf("Invalid choice!\n");

}

} while(choice != 3);

return 0; }

**Output -:**

****

**Q. Count Leaf Nodes in Binary Search Tree**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* insert(struct Node\* root, int data) {

if (root == NULL) {

root = (struct Node\*)malloc(sizeof(struct Node));

root->data = data;

root->left = root->right = NULL;

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

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

} else {

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

}

return root;

}

int countLeafNodes(struct Node\* root) {

if (root == NULL) return 0;

if (root->left == NULL && root->right == NULL) return 1;

return countLeafNodes(root->left) + countLeafNodes(root->right);

}

int main() {

struct Node\* root = NULL;

int choice, value;

do {

printf("\n1. Insert Node\n2. Count Leaf Nodes\n3. Exit\nEnter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

root = insert(root, value);

break;

case 2:

printf("Leaf Nodes: %d\n", countLeafNodes(root));

break;

case 3:

printf("Exiting...\n");

break;

default:

printf("Invalid choice!\n");

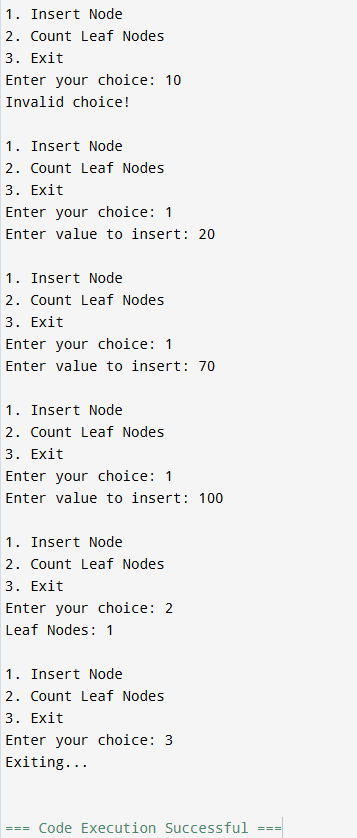
}

} while(choice != 3);

return 0;

}

**Output :**

****

**Q. Count Nodes Having One Child in Binary Search Tree**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* insert(struct Node\* root, int data) {

if (root == NULL) {

root = (struct Node\*)malloc(sizeof(struct Node));

root->data = data;

root->left = root->right = NULL;

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

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

} else {

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

}

return root;

}

int countOneChild(struct Node\* root) {

if (root == NULL) return 0;

int count = 0;

if ((root->left == NULL && root->right != NULL) || (root->left != NULL && root->right == NULL)) {

count = 1;

}

return count + countOneChild(root->left) + countOneChild(root->right);

}

int main() {

struct Node\* root = NULL;

int choice, value;

do {

printf("\n1. Insert Node\n2. Count Nodes With One Child\n3. Exit\nEnter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

root = insert(root, value);

break;

case 2:

printf("Nodes with One Child: %d\n", countOneChild(root));

break;

case 3:

printf("Exiting...\n");

break;

default:

printf("Invalid choice!\n");

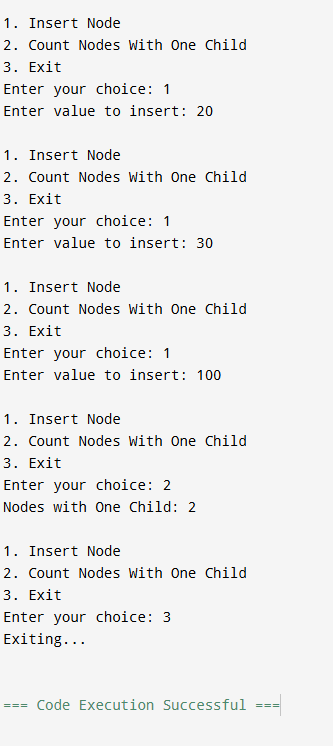
}

} while(choice != 3);

return 0;

}

**Output :**

****

**Q. Count Nodes Having Only Left Child**

**#include <stdio.h>**

**#include <stdlib.h>**

**struct Node {**

**int data;**

**struct Node\* left;**

**struct Node\* right;**

**};**

**struct Node\* insert(struct Node\* root, int data) {**

**if (root == NULL) {**

**root = (struct Node\*)malloc(sizeof(struct Node));**

**root->data = data;**

**root->left = root->right = NULL;**

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

**root->left = insert(root->left, data);**

**} else {**

**root->right = insert(root->right, data);**

**}**

**return root;**

**}**

**int countLeftOnly(struct Node\* root) {**

**if (root == NULL) return 0;**

**int count = 0;**

**if (root->left != NULL && root->right == NULL) {**

**count = 1;**

**}**

**return count + countLeftOnly(root->left) + countLeftOnly(root->right);**

**}**

**int main() {**

**struct Node\* root = NULL;**

**int choice, value;**

**do {**

**printf("\n1. Insert Node\n2. Count Nodes With Only Left Child\n3. Exit\nEnter your choice: ");**

**scanf("%d", &choice);**

**switch(choice) {**

**case 1:**

**printf("Enter value to insert: ");**

**scanf("%d", &value);**

**root = insert(root, value);**

**break;**

**case 2:**

**printf("Nodes with Only Left Child: %d\n", countLeftOnly(root));**

**break;**

**case 3:**

**printf("Exiting...\n");**

**break;**

**default:**

**printf("Invalid choice!\n");**

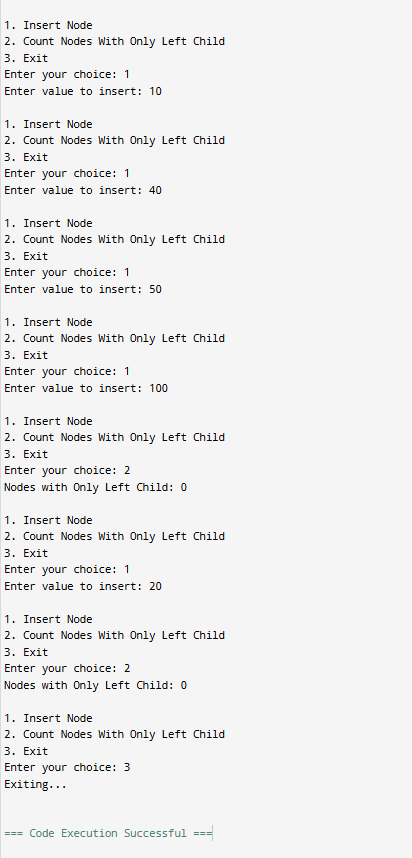
**}**

**} while(choice != 3);**

**return 0;**

**}**

**Output:**

****