Q-1. Write a program to do the following operations.

● Create a Binary Tree by collecting information from users.

● Create a Binary Search Tree by collecting information from users.

● Traverse the created trees using

○ preorder

○ postorder

○ inorder

○ level order

● Search Element in Binary Search Tree

● Find Internal Nodes, External Nodes, Total Nodes and Height of Tree

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;struct Node\* left;

struct Node\* right;

}

struct Node\* create(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node)); newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

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

if (root == NULL) {

return create(data);

}

if (data < root->data) {

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

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

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

}

return root;

}

struct Node\* createBinaryTree() {

int data;

struct Node\* root = NULL;

printf("Enter root node data: ");

scanf("%d", &data);

root = create(data);

printf("Enter left child of %d : ", data);

scanf("%d", &data);

if (data != -1) {

root->left = create(data);

printf("Enter left child of %d : ", root->left->data);

scanf("%d", &data);

if (data != -1) {

root->left->left = create(data);

}

printf("Enter right child of %d : ", root->left->data);

scanf("%d", &data);

if (data != -1) {

root->left->right = create(data);

}

}

printf("Enter right child of %d : ", data);

scanf("%d", &data);

if (data != -1) {

root->right = create(data);

printf("Enter left child of %d : ", root->right->data);

scanf("%d", &data);

if (data != -1) {

root->right->left = create(data);

}

printf("Enter right child of %d : ", root->right->data);

scanf("%d", &data);

if (data != -1) {

root->right->right = create(data);

}

}

return root;

}

void preorderTraversal(struct Node\* root) {

if (root == NULL)

return;

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

preorderTraversal(root->left);

preorderTraversal(root->right);

}

void inorderTraversal(struct Node\* root) {

if (root == NULL)

return;

inorderTraversal(root->left);

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

inorderTraversal(root->right);

}

void postorderTraversal(struct Node\* root) {

if (root == NULL)

return;

postorderTraversal(root->left);

postorderTraversal(root->right);

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

}

void levelorderTraversal(struct Node\* root) {

if (root == NULL)

return;

struct Node\*\* queue = (struct Node\*\*)malloc(sizeof(struct Node\*) \* 100);

int front = -1;

int rear = -1;

queue[++rear] = root;

while (front < rear) {

struct Node\* node = queue[++front];

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

if (node->left != NULL)

queue[++rear] = node->left;

if (node->right != NULL)

queue[++rear] = node->right;

}

}

void countNodesAndHeight(struct Node\* root, int\* internalNodes, int\* externalNodes, int\* totalNodes, int\* height) {

if (root == NULL)

return;

(\*totalNodes)++;

if (root->left == NULL && root->right == NULL)

(\*externalNodes)++;

else

(\*internalNodes)++;

int leftHeight = 0, rightHeight = 0;

countNodesAndHeight(root->left, internalNodes, externalNodes, totalNodes, &leftHeight);

countNodesAndHeight(root->right, internalNodes, externalNodes, totalNodes, &rightHeight);

(\*height) = 1 + (leftHeight > rightHeight ? leftHeight : rightHeight);

}

struct Node\* searchNode(struct Node\* root, int key) {

if (root == NULL || root->data == key)

return root;

if (root->data < key)

return searchNode(root->right, key);

else

return searchNode(root->left, key);

}

int main() {

struct Node\* root = NULL;

int choice, data, key, internalNodes = 0, externalNodes = 0, totalNodes = 0, height = 0;

do {

printf("\n--- Binary Tree and Binary Search Tree Operations ---\n");

printf("1. Create Binary Tree\n");

printf("2. Create Binary Search Tree\n");

printf("3. Preorder Traversal\n");

printf("4. Inorder Traversal\n");

printf("5. Postorder Traversal\n");

printf("6. Levelorder Traversal\n");

printf("7. Search Element in Binary Search Tree\n");

printf("8. Count Internal Nodes, External Nodes, Total Nodes, and Height of Tree\n"); printf("9. Exit\n");

printf("Enter your choice: "); scanf("%d", &choice);

switch (choice) {

case 1:

root = createBinaryTree();

break;

case 2:

printf("Enter root node data: ");

scanf("%d", &data);

root = create(data);

while (1) {

printf("Enter data to be inserted: in stop then -1 ");

scanf("%d", &data);

if (data == -1)

break;

insertNode(root, data);

}

break;

case 3:

printf("Preorder Traversal: ");

preorderTraversal(root);

break;

case 4:

printf("Inorder Traversal: ");

inorderTraversal(root);

break;

case 5:

printf("Postorder Traversal: ");

postorderTraversal(root);

break;

case 6:

printf("Levelorder Traversal: ");

levelorderTraversal(root);

break;

case 7:

printf("Enter element to search: ");

scanf("%d", &key);

if (searchNode(root, key) != NULL)

printf("Element found in the tree.\n");

else

printf("Element not found in the tree.\n");

break;

case 8:

countNodesAndHeight(root, &internalNodes, &externalNodes, &totalNodes, &height);

printf("Total number of nodes in the tree: %d\n", totalNodes);

printf("Number of internal nodes in the tree: %d\n", internalNodes);

printf("Number of external nodes in the tree: %d\n", externalNodes);

printf("Height of the tree: %d\n", height);

break;

case 9:

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

exit(0);

default:

printf("Invalid choice!\n");

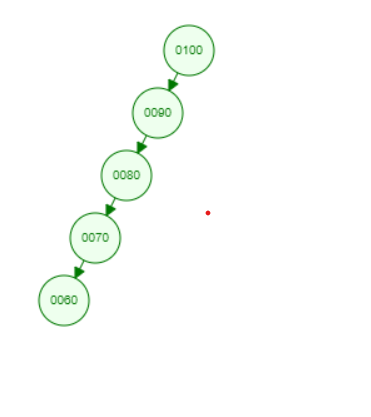
break;

}

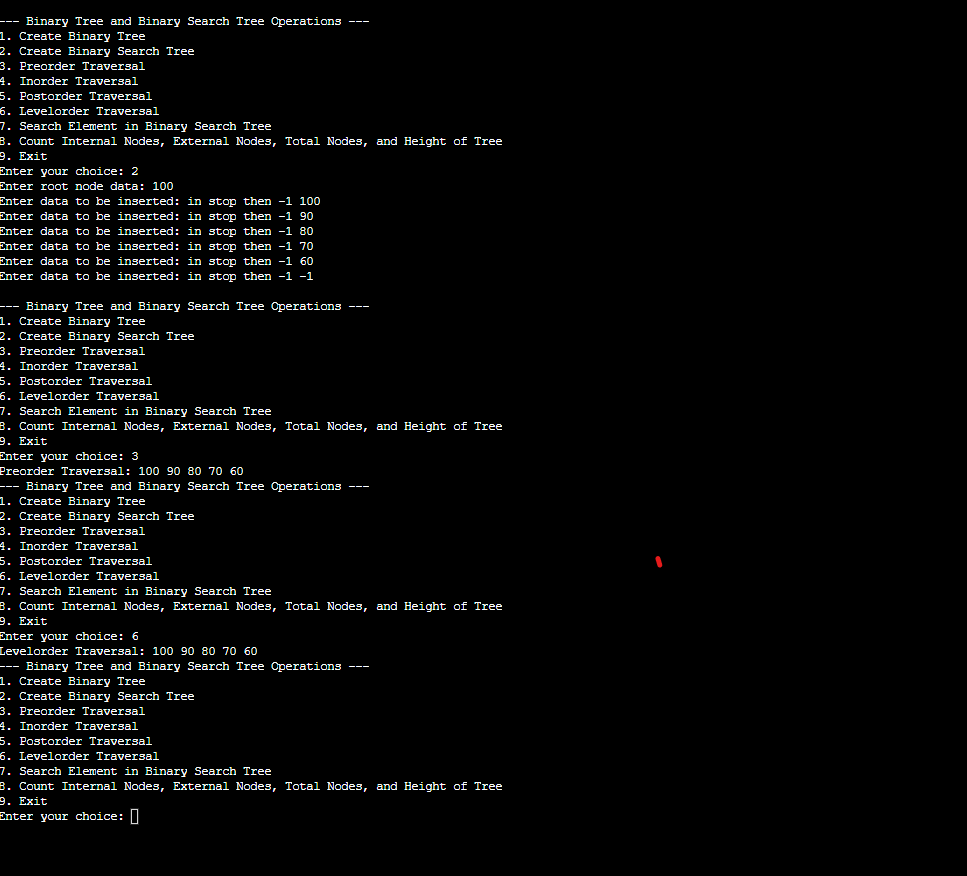
} while (choice != 9);

return 0;

}



**OUTPUT:**



**Question 2:**

2. Write a program to do the following operations.

● Create an array from user input.

● Search Element in an array using linear search - prints iteration done to find the element

● Search Element in an array using binary search - prints iteration done to find the element

**Source Code:**

#include <stdio.h>

int LinearSearch(int arr[],int size,intkey)

{

for(int i=0; i<size; i++){

if(arr[i]==key)

return i;

}

return -1;

}

int BinarySearch(int arr[],int size,int key){ int start=0;

int end= size-1;

while(start<= end){

int mid= start+(end-start)/2;

if(arr[mid]==key)

return mid;

else if(arr[mid] > key)

end=mid-1;

else

start=mid+1;

}

return start;

}

int main(){

int size;

printf("Enter the Size of Arrays : \n"); scanf("%d",&size);

int arr[size];

scanf("%d",&arr[i]);

}

printf("Enter the Value in to Arrays : \n");

for(int i=0; i<size; i++){

scanf("%d",&arr[i]);

}

int key;

printf("Enter the Key to Want Search : ");

scanf("%d",&key);

int n;

printf("Enter the Number to Perform Perticular Searching Algorithm \n Zero (0) For Linear Search \n One

(1) For Binary Search\n");

scanf("%d",&n);

switch(n){

case 0:

printf("Value will be available on This Index : %d\n",LinearSearch(arr,size,key));

break;

case 1:

printf("Value will be available on This Index : %d\n",BinarySearch(arr,size,key));

break;

default:

printf("You select wrong number : ");

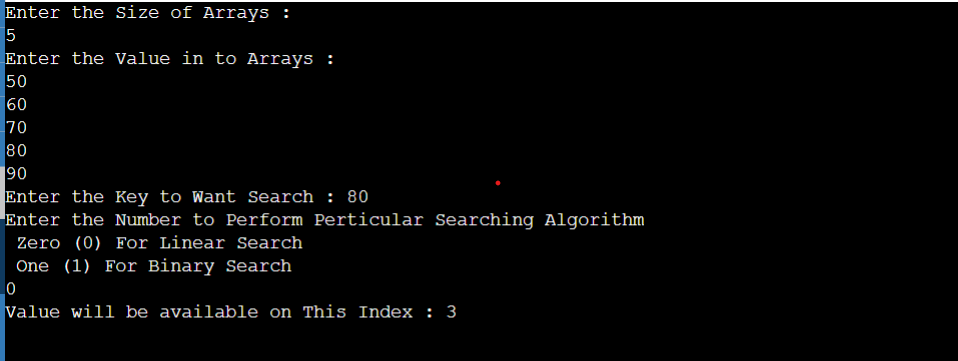
break;

}

return 0;

}

**Output:**

****