**DSC LAB – {8}**

Akshaya agarwal

2247207

3-MCA B

**CODE :**

**#include <stdio.h>**

**#include <stdlib.h>**

**// structure of a node in the binary search tree**

**struct node {**

**int data;**

**struct node \*left;**

**struct node \*right;**

**};**

**// function prototypes**

**struct node \*create\_node(int value);**

**struct node \*insert(struct node \*root, int value);**

**struct node \*delete(struct node \*root, int value);**

**struct node \*min\_value\_node(struct node \*node);**

**void inorder\_traversal(struct node \*root);**

**void preorder\_traversal(struct node \*root);**

**void postorder\_traversal(struct node \*root);**

**int main() {**

**struct node \*root = NULL;**

**int choice, value;**

**do {**

**// print menu**

**printf("\nBinary Search Tree Menu\n");**

**printf("-----------------------\n");**

**printf("1. Insert\n");**

**printf("2. Delete\n");**

**printf("3. Search\n");**

**printf("4. Inorder traversal\n");**

**printf("5. Preorder traversal\n");**

**printf("6. Postorder traversal\n");**

**printf("7. Exit\n");**

**// get user choice**

**printf("\nEnter your choice (1-7): ");**

**// read input and validate that it is a number**

**char input[100];**

**int valid = 0;**

**while (!valid) {**

**if (fgets(input, 100, stdin) != NULL) {**

**if (sscanf(input, "%d", &choice) == 1) {**

**valid = 1;**

**}**

**}**

**if (!valid) {**

**printf("Invalid choice. Please enter a value between 1 and 7: ");**

**}**

**}**

**switch (choice) {**

**case 1:**

**// insert node**

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

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

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

**break;**

**case 2:**

**// delete node**

**printf("Enter the value to delete: ");**

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

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

**break;**

**case 3:**

**// search for node**

**printf("Enter the value to search for: ");**

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

**if (search(root, value)) {**

**printf("Value %d was found in the tree.\n", value);**

**} else {**

**printf("Value %d was not found in the tree.\n", value);**

**}**

**break;**

**case 4:**

**// inorder traversal**

**printf("Inorder traversal: ");**

**inorder\_traversal(root);**

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

**break;**

**case 5:**

**// preorder traversal**

**printf("Preorder traversal: ");**

**preorder\_traversal(root);**

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

**break;**

**case 6:**

**// postorder traversal**

**printf("Postorder traversal: ");**

**postorder\_traversal(root);**

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

**break;**

**case 7:**

**// exit program**

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

**break;**

**default:**

**printf("Invalid choice. Please enter a value between 1 and 7.\n");**

**break;**

**}**

**} while (choice != 7);**

**return 0;**

**}**

**// create a new node with the given value**

**struct node \*create\_node(int value) {**

**struct node \*new\_node = (struct node \*) malloc(sizeof(struct node));**

**new\_node->data = value;**

**new\_node->left = NULL;**

**new\_node->right = NULL;**

**return new\_node;**

**}**

**// insert a new node with the given value into the tree**

**struct node \*insert(struct node \*root, int value) {**

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

**return create\_node(value);**

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

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

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

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

**}**

**return root;**

**}**

**// search for a node with the given value**

**// in the tree**

**int search(struct node \*root, int value) {**

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

**return 0;**

**} else if (value == root->data) {**

**return 1;**

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

**return search(root->left, value);**

**} else {**

**return search(root->right, value);**

**}**

**}**

**// delete a node with the given value from the tree**

**struct node \*delete(struct node \*root, int value) {**

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

**return NULL;**

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

**root->left = delete(root->left, value);**

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

**root->right = delete(root->right, value);**

**} else {**

**// node with value found**

**if (root->left == NULL) {**

**// node has no left child**

**struct node \*temp = root->right;**

**free(root);**

**return temp;**

**} else if (root->right == NULL) {**

**// node has no right child**

**struct node \*temp = root->left;**

**free(root);**

**return temp;**

**} else {**

**// node has two children**

**struct node \*temp = min\_value\_node(root->right);**

**root->data = temp->data;**

**root->right = delete(root->right, temp->data);**

**}**

**}**

**return root;**

**}**

**// find the node with the minimum value in the tree**

**struct node \*min\_value\_node(struct node \*node) {**

**struct node \*current = node;**

**while (current->left != NULL) {**

**current = current->left;**

**}**

**return current;**

**}**

**// perform an inorder traversal of the tree**

**void inorder\_traversal(struct node \*root) {**

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

**inorder\_traversal(root->left);**

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

**inorder\_traversal(root->right);**

**}**

**}**

**// perform a preorder traversal of the tree**

**void preorder\_traversal(struct node \*root) {**

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

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

**preorder\_traversal(root->left);**

**preorder\_traversal(root->right);**

**}**

**}**

**// perform a postorder traversal of the tree**

**void postorder\_traversal(struct node \*root) {**

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

**postorder\_traversal(root->left);**

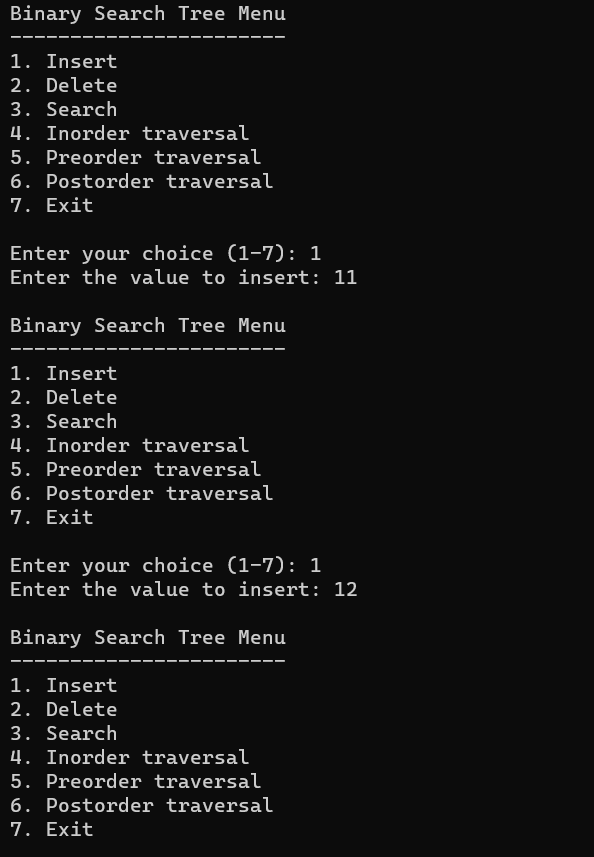
**postorder\_traversal(root->right);**

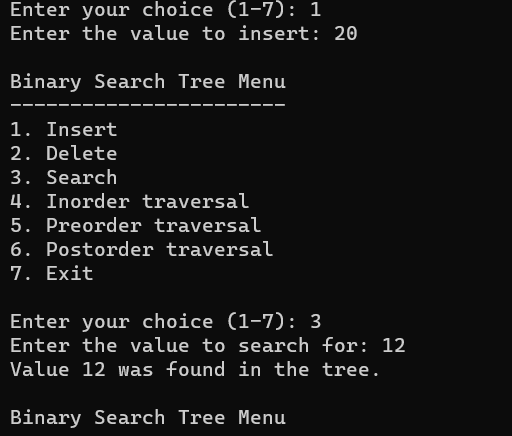
**printf("%d ", root->data);**

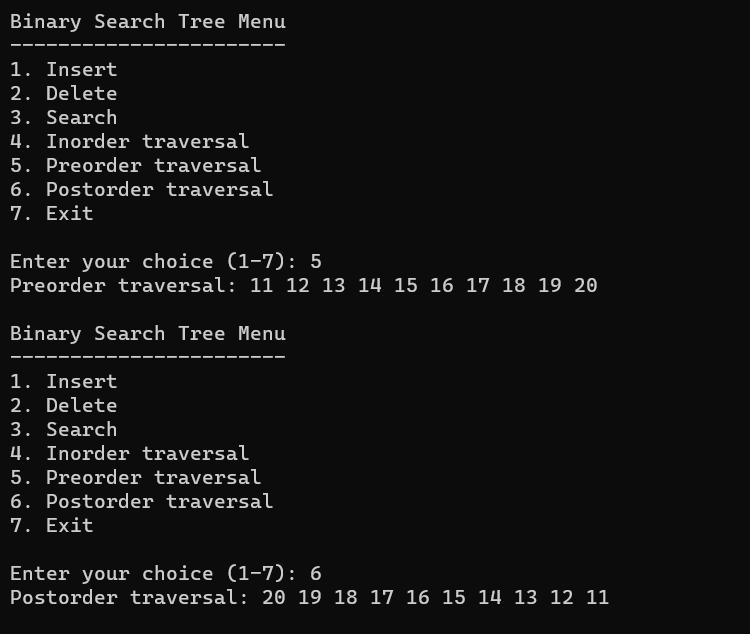
**}**

**}**

}**OUTPUT :**

****

****

****