```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node *left, *right;
};

struct Node* newNode(int data) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;
    node->left = node->right = NULL;
    return node;
}

struct Node* insert(struct Node* node, int data) {
    if (node == NULL) {
        return newNode(data);
    }

    if (data < node->data) {
        node->left = insert(node->left, data);
    } else if (data > node->data) {
        node->right = insert(node->right, data);
    }

    return node;
}

struct Node* minValueNode(struct Node* node) {
    struct Node* current = node;
    while (current && current->left != NULL) {
        current = current->left;
    }
    return current;
}
```

```c
int search(struct Node* root, int data) {
    if (root == NULL) {
        return 0;
    }

    if (root->data == data) {
        return 1;
    } else if (data < root->data) {
        return search(root->left, data);
    } else {
        return search(root->right, data);
    }
}

int main() {
    struct Node* root = NULL;
    int choice, data;

    while (1) {
        printf("\n1. Insert\n");
        printf("2. Delete\n");
        printf("3. Inorder Traversal\n");
        printf("4. Preorder Traversal\n");
        printf("5. Postorder Traversal\n");
        printf("6. Search\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                root = insert(root, data);
                break;

            case 2:
                printf("Enter data to delete: ");
                scanf("%d", &data);
                root = deleteNode(root, data);
                break;

            case 3:
                printf("Inorder Traversal: ");
                inorderTraversal(root);
                printf("\n");
                break;
```

```c
struct Node* deleteNode(struct Node* root, int data) {
    if (root == NULL) {
        return root;
    }

    if (data < root->data) {
        root->left = deleteNode(root->left, data);
    } else if (data > root->data) {
        root->right = deleteNode(root->right, data);
    } else {
        if (root->left == NULL) {
            struct Node* temp = root->right;
            free(root);
            return temp;
        } else if (root->right == NULL) {
            struct Node* temp = root->left;
            free(root);
            return temp;
        }

        struct Node* temp = minValueNode(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
}
void inorderTraversal(struct Node* root) {
    if (root != NULL) {
        inorderTraversal(root->left);
        printf("%d ", root->data);
        inorderTraversal(root->right);
    }
}
void preorderTraversal(struct Node* root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorderTraversal(root->left);
        preorderTraversal(root->right);
    }
}
void postorderTraversal(struct Node* root) {
    if (root != NULL) {
        postorderTraversal(root->left);
        postorderTraversal(root->right);
        printf("%d ", root->data);
    }
}
```

```c
            case 4:
                printf("Preorder Traversal: ");
                preorderTraversal(root);
                printf("\n");
                break;

            case 5:
                printf("Postorder Traversal: ");
                postorderTraversal(root);
                printf("\n");
                break;

            case 6:
                printf("Enter data to search: ");
                scanf("%d", &data);
                if (search(root, data)) {
                    printf("Element found\n");
                } else {
                    printf("Element not found\n");
                }
                break;

            case 7:
                exit(0);

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

    return 0;
}
```