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

struct node {
  int data;
  struct node *right_child;
  struct node *left_child;
};

struct node* new_node(int x){
  struct node *temp;
  temp = malloc(sizeof(struct node));
  temp->data = x;
  temp->left_child = NULL;
  temp->right_child = NULL;

  return temp;
}

struct node* search(struct node * root, int x){
  if (root == NULL || root->data == x)
    return root;
  else if (x > root->data)
    return search(root->right_child, x);
  else
    return search(root->left_child, x);
}

struct node* insert(struct node * root, int x){
  if (root == NULL)
    return new_node(x);
  else if (x > root->data)
    root->right_child = insert(root->right_child, x);
  else
    root -> left_child = insert(root->left_child, x);
  return root;
}
struct node* find_minimum(struct node * root) {
  if (root == NULL)
    return NULL;
  else if (root->left_child != NULL)
    return find_minimum(root->left_child);
  return root;
}

struct node* delete(struct node * root, int x) {
```

```
  if (root == NULL)
    return NULL;
  if (x > root->data)
    root->right_child = delete(root->right_child, x);
  else if (x < root->data)
    root->left_child = delete(root->left_child, x);
  else {
    if (root->left_child == NULL && root->right_child == NULL){
      free(root);
      return NULL;
    }
    else if (root->left_child == NULL || root->right_child == NULL){
      struct node *temp;
      if (root->left_child == NULL)
        temp = root->right_child;
      else
        temp = root->left_child;
      free(root);
      return temp;
    }
    else {
      struct node *temp = find_minimum(root->right_child);
      root->data = temp->data;
      root->right_child = delete(root->right_child, temp->data);
    }
  }
  return root;
}

void inorder(struct node *root){
  if (root != NULL)
  {
    inorder(root->left_child);
    printf(" %d ", root->data);
    inorder(root->right_child);
  }
}

int main() {
    struct node* root = NULL;  // Start with an empty tree
    int choice, value;

    while (1) {
        printf("\nBinary Search Tree Operations:\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Inorder Traversal\n");
```

```
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                root = insert(root, value);
                printf("Inserted %d into the BST.\n", value);
                break;
            case 2:
                printf("Enter value to delete: ");
                scanf("%d", &value);
                root = delete(root, value);
                printf("Deleted %d from the BST.\n", value);
                break;
            case 3:
                printf("Inorder Traversal: ");
                inorder(root);
                printf("\n");
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```