Experiment 20: Binary Search Tree

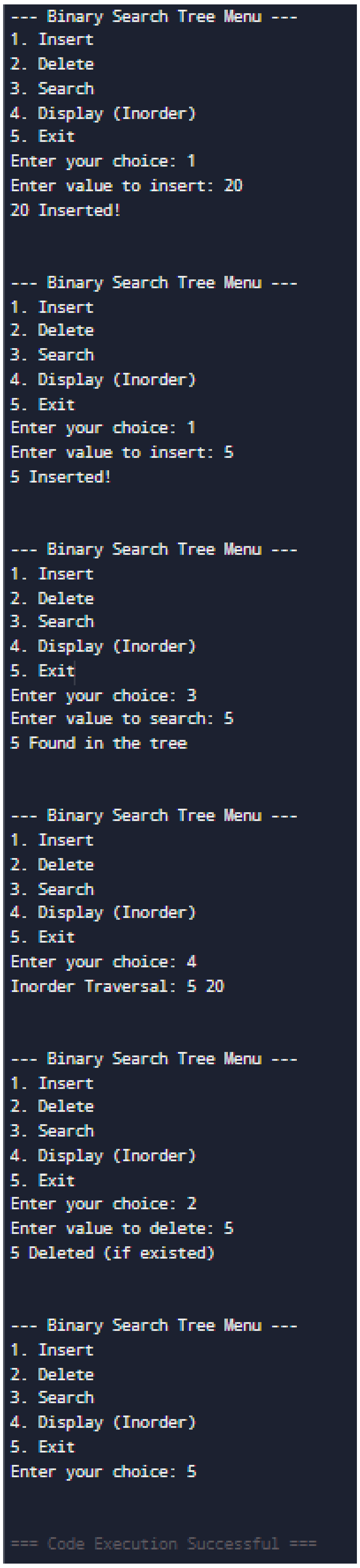
Code:

```
#include  <stdio.h>
#include <stdlib.h>
struct node {
   int data;
   struct node *left, *right;
};
struct node* createNode(int value) {
   struct node* newNode = (struct node*)malloc(sizeof(struct node));
   newNode->data = value;
   newNode->left = newNode->right = NULL;
   return newNode;
}
struct node* insert(struct node* root, int value) {
   if (root == NULL)
      return createNode(value);
   if (value < root->data)
      root->left = insert(root->left, value);
   else if (value > root->data)
      root->right = insert(root->right, value);
   return root;
}
struct node* search(struct node* root, int key) {
   if (root == NULL || root->data == key)
      return root;
   if (key < root->data)
```

```
        return search(root->left, key);
    else
        return search(root->right, key);
}
struct node* findMin(struct node* root) {
    while (root->left != NULL)
        root = root->left;
    return root;
}
struct node* deleteNode(struct node* root, int key) {
    if (root == NULL) return root;
    if (key < root->data)
        root->left = deleteNode(root->left, key);
    else if (key > root->data)
        root->right = deleteNode(root->right, key);
    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 = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
```

```
}
void inorder(struct node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}
int main() {
    struct node* root = NULL;
    int choice, value;
    while (1) {
        printf("\n\n--- Binary Search Tree Menu ---\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Search\n");
        printf("4. Display (Inorder)\n");
        printf("5. 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("%d Inserted!\n", value);
                break;
            case 2:
                printf("Enter value to delete: ");
                scanf("%d", &value);
                root = deleteNode(root, value);
                printf("%d Deleted (if existed)\n", value);
                break;
            case 3:
                printf("Enter value to search: ");
                scanf("%d", &value);
                if (search(root, value) != NULL)
                    printf("%d Found in the tree\n", value);
                else
                    printf("%d Not found!\n", value);
                break;
            case 4:
                printf("Inorder Traversal: ");
                inorder(root);
                printf("\n");
                break;
            case 5:
                exit(0);
            default:
                printf("Invalid choice. Try again!\n");
        }
    }
}
```

Output:



```
--- Binary Search Tree Menu ---
1. Insert
2. Delete
3. Search
4. Display (Inorder)
5. Exit
Enter your choice: 1
Enter value to insert: 20
20 Inserted!


--- Binary Search Tree Menu ---
1. Insert
2. Delete
3. Search
4. Display (Inorder)
5. Exit
Enter your choice: 1
Enter value to insert: 5
5 Inserted!


--- Binary Search Tree Menu ---
1. Insert
2. Delete
3. Search
4. Display (Inorder)
5. Exit
Enter your choice: 3
Enter value to search: 5
5 Found in the tree


--- Binary Search Tree Menu ---
1. Insert
2. Delete
3. Search
4. Display (Inorder)
5. Exit
Enter your choice: 4
Inorder Traversal: 5 20


--- Binary Search Tree Menu ---
1. Insert
2. Delete
3. Search
4. Display (Inorder)
5. Exit
Enter your choice: 2
Enter value to delete: 5
5 Deleted (if existed)


--- Binary Search Tree Menu ---
1. Insert
2. Delete
3. Search
4. Display (Inorder)
5. Exit
Enter your choice: 5


=== Code Execution Successful ===
```