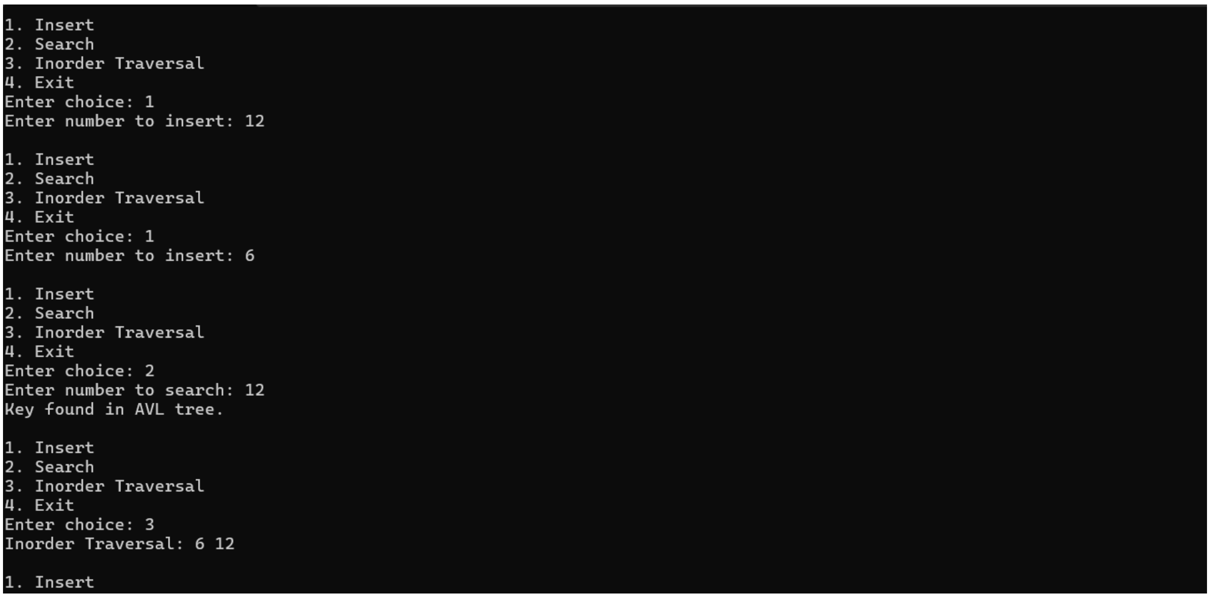
AVL Tree

CODE:

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int key;
    struct Node *left, *right;
    int height;
};
int height(struct Node *n) {
    return (n == NULL) ? 0 : n->height;
}
int max(int a, int b) {
    return (a > b) ? a : b;
}
struct Node* newNode(int key) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->key = key;
    node->left = node->right = NULL;
    node->height = 1;
    return node;
}
struct Node* rightRotate(struct Node* y) {
    struct Node* x = y->left;
    struct Node* T2 = x->right;
    x->right = y;
    y->left = T2;
    y->height = max(height(y->left), height(y->right)) + 1;
    x->height = max(height(x->left), height(x->right)) + 1;
    return x;
```

```
}
struct Node* leftRotate(struct Node* x) {
    struct Node* y = x->right;
    struct Node* T2 = y->left;
    y->left = x;
    x->right = T2;
    x->height = max(height(x->left), height(x->right)) + 1;
    y->height = max(height(y->left), height(y->right)) + 1;
    return y;
}
int getBalance(struct Node* n) {
    return (n == NULL) ? 0 : height(n->left) - height(n->right);
}
struct Node* insert(struct Node* node, int key) {
    if (node == NULL)
        return newNode(key);
    if (key < node->key)
        node->left = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);
    else
        return node;
    node->height = 1 + max(height(node->left), height(node->right));
    int balance = getBalance(node);
    if (balance > 1 && key < node->left->key)
        return rightRotate(node);
    if (balance < -1 && key > node->right->key)
        return leftRotate(node);
    if (balance > 1 && key > node->left->key) {
        node->left = leftRotate(node->left);
        return rightRotate(node);
    }
    if (balance < -1 && key < node->right->key) {
        node->right = rightRotate(node->right);
        return leftRotate(node);
    }
    return node;
}
void inorder(struct Node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->key);
        inorder(root->right);
    }
}
int search(struct Node* root, int key) {
    if (root == NULL)
        return 0;
    if (root->key == key)
        return 1;
    else if (key < root->key)
        return search(root->left, key);
    else
        return search(root->right, key);
}
int main() {
    struct Node* root = NULL;
    int choice, key;
    while (1) {
        printf("\n1. Insert\n2. Search\n3. Inorder Traversal\n4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter number to insert: ");
                scanf("%d", &key);
                root = insert(root, key);
                break;
            case 2:
                printf("Enter number to search: ");
                scanf("%d", &key);
                if (search(root, key))
                    printf("Key found in AVL tree.\n");
                else
                    printf("Key not found.\n");
                break;
            case 3:
                printf("Inorder Traversal: ");
                inorder(root);
                printf("\n");
                break;
            case 4:
                return 0;
            default:
                printf("Invalid choice.\n");
        }
    }
    return 0;
}
```

OUTPUT:



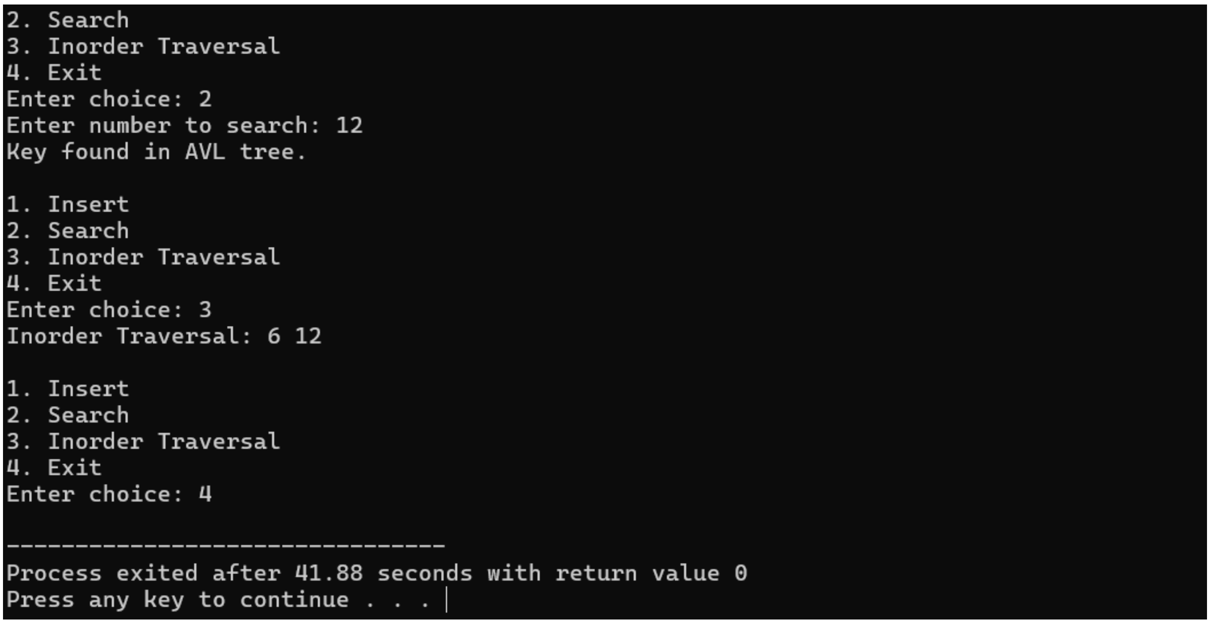
```
1. Insert
2. Search
3. Inorder Traversal
4. Exit
Enter choice: 1
Enter number to insert: 12

1. Insert
2. Search
3. Inorder Traversal
4. Exit
Enter choice: 1
Enter number to insert: 6

1. Insert
2. Search
3. Inorder Traversal
4. Exit
Enter choice: 2
Enter number to search: 12
Key found in AVL tree.

1. Insert
2. Search
3. Inorder Traversal
4. Exit
Enter choice: 3
Inorder Traversal: 6 12

1. Insert
```



```
2. Search
3. Inorder Traversal
4. Exit
Enter choice: 2
Enter number to search: 12
Key found in AVL tree.

1. Insert
2. Search
3. Inorder Traversal
4. Exit
Enter choice: 3
Inorder Traversal: 6 12

1. Insert
2. Search
3. Inorder Traversal
4. Exit
Enter choice: 4

--------------------------------
Process exited after 41.88 seconds with return value 0
Press any key to continue . . .
```