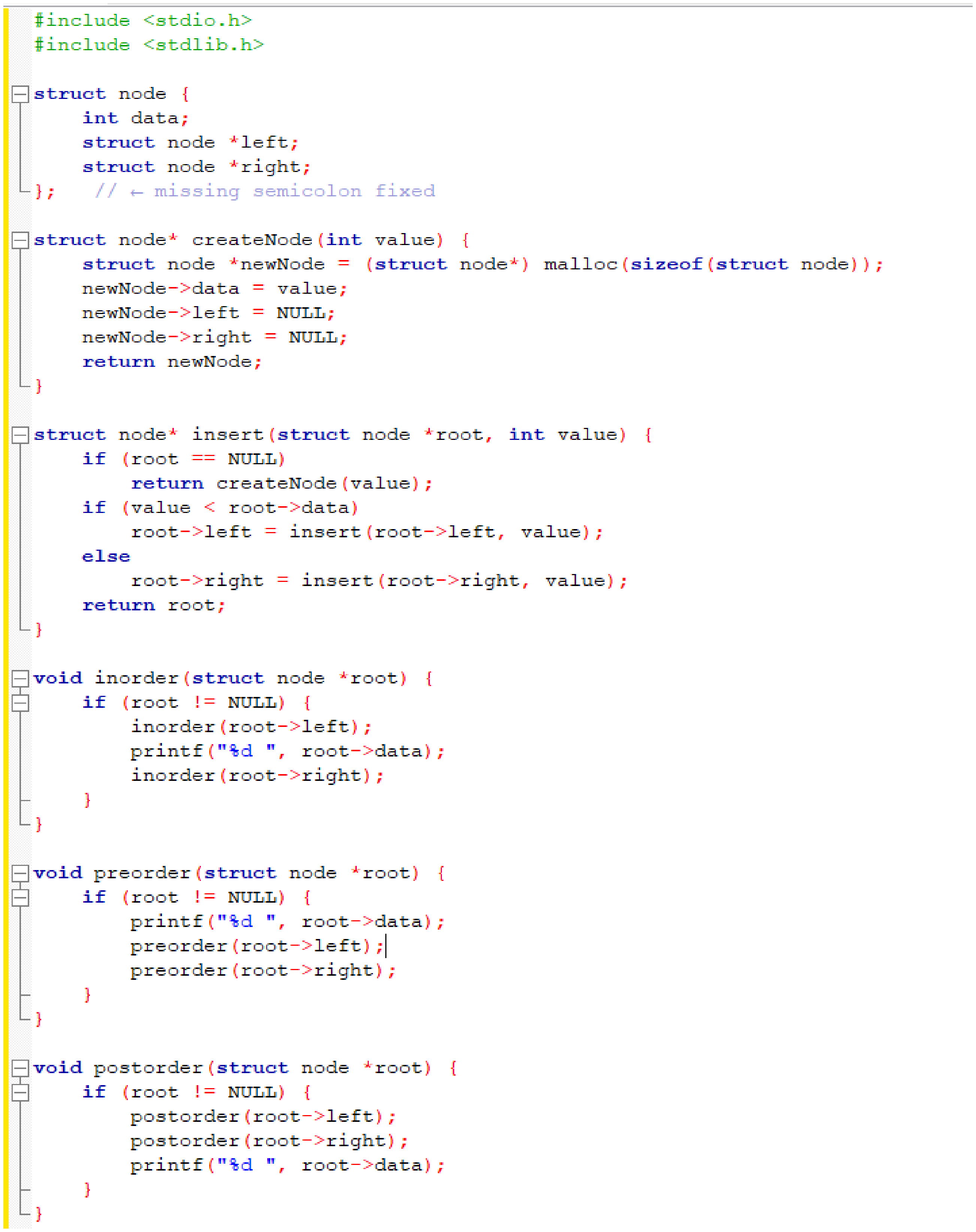
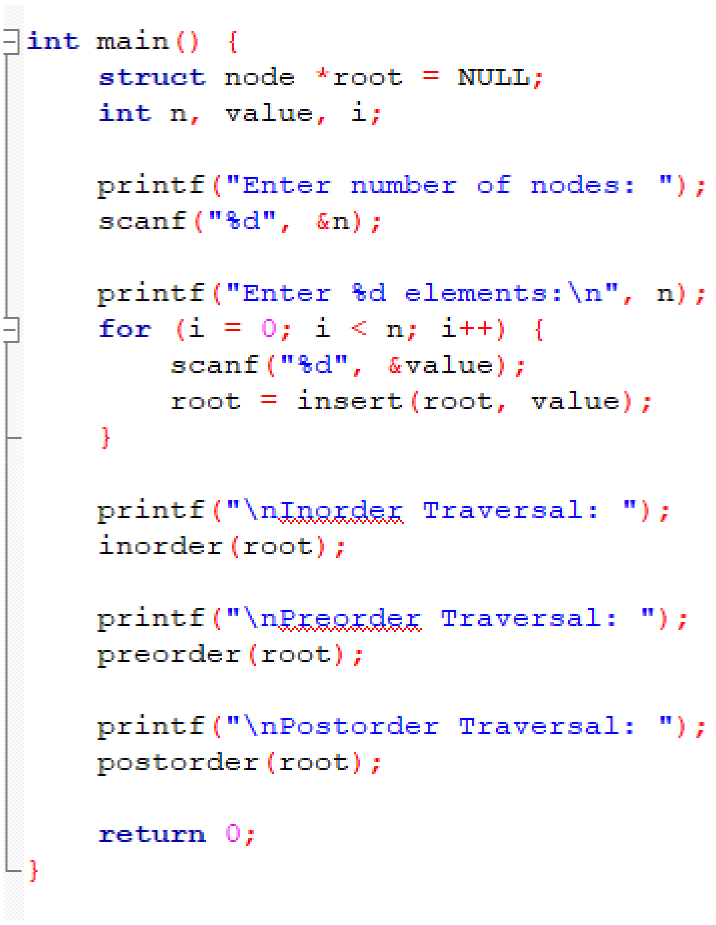
WEEK—9



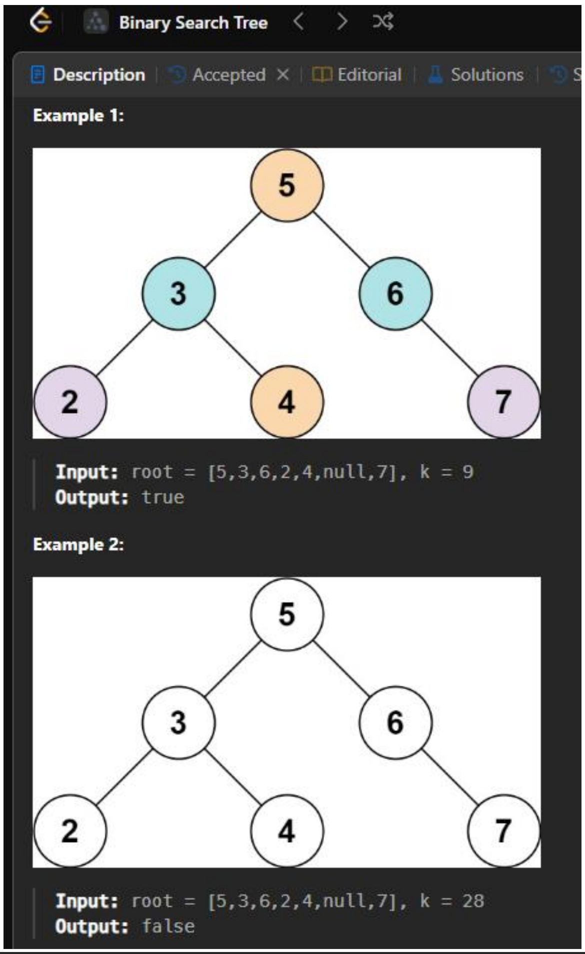
```c
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *left;
    struct node *right;
};   // ← missing semicolon fixed

struct node* createNode(int value) {
    struct node *newNode = (struct node*) malloc(sizeof(struct node));
    newNode->data = value;
    newNode->left = NULL;
    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
        root->right = insert(root->right, value);
    return root;
}

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

void preorder(struct node *root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

void postorder(struct node *root) {
    if (root != NULL) {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
}
```



```c
int main() {
    struct node *root = NULL;
    int n, value, i;

    printf("Enter number of nodes: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &value);
        root = insert(root, value);
    }

    printf("\nInorder Traversal: ");
    inorder(root);

    printf("\nPreorder Traversal: ");
    preorder(root);

    printf("\nPostorder Traversal: ");
    postorder(root);

    return 0;
}
```

```
Enter number of nodes: 5
Enter 5 elements:
23
54
1
62
1

Inorder Traversal: 1 1 23 54 62
Preorder Traversal: 23 1 1 54 62
Postorder Traversal: 1 1 62 54 23
```

LEET CODE PROGRAM

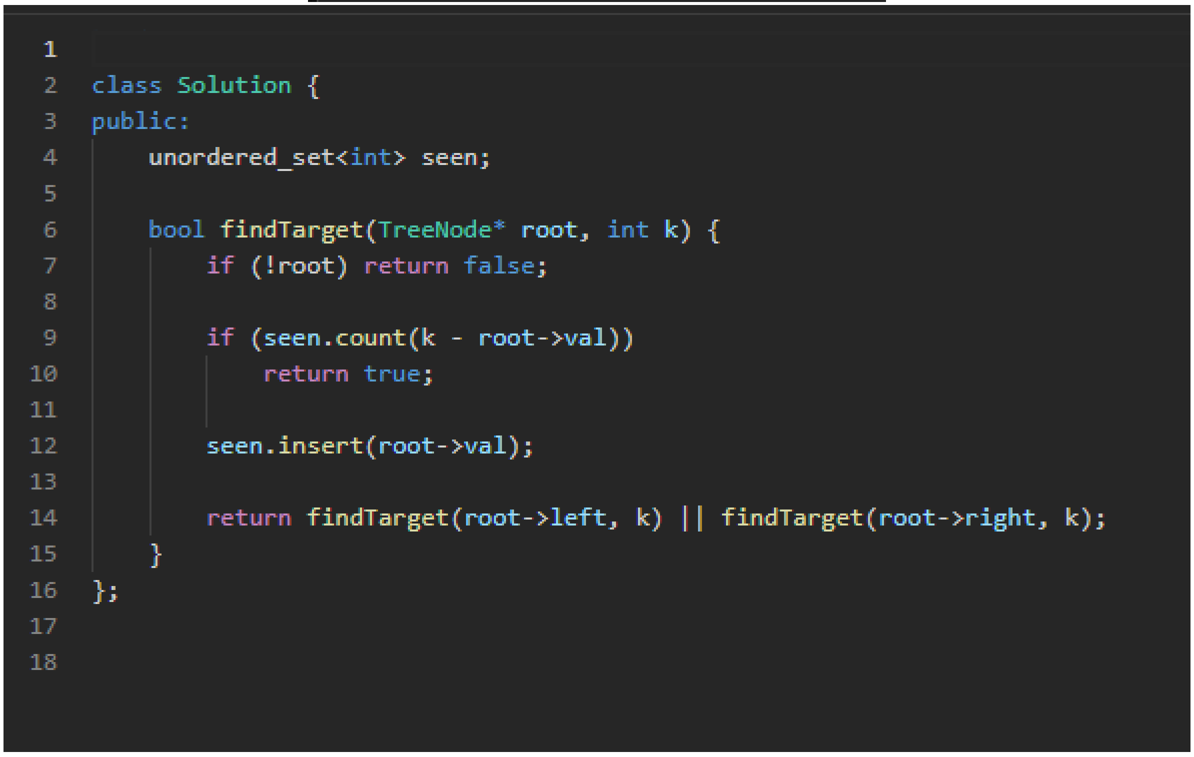


Example 1:

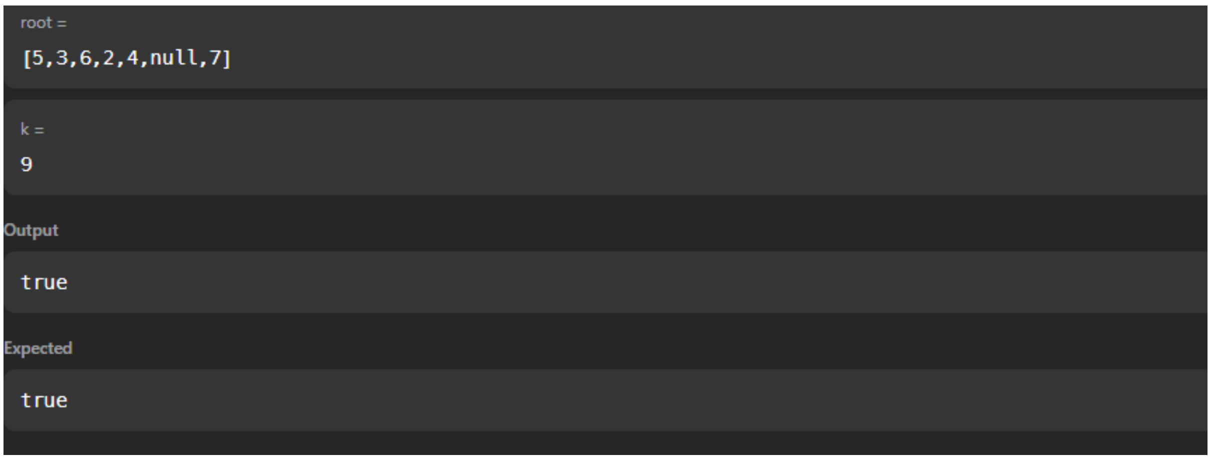
```
Input: root = [5,3,6,2,4,null,7], k = 9
Output: true
```

Example 2:

```
Input: root = [5,3,6,2,4,null,7], k = 28
Output: false
```



```

class Solution {
public:
    unordered_set<int> seen;

    bool findTarget(TreeNode* root, int k) {
        if (!root) return false;

        if (seen.count(k - root->val))
            return true;

        seen.insert(root->val);

        return findTarget(root->left, k) || findTarget(root->right, k);
    }
};


```



root =

```
[5,3,6,2,4,null,7]
```

k =

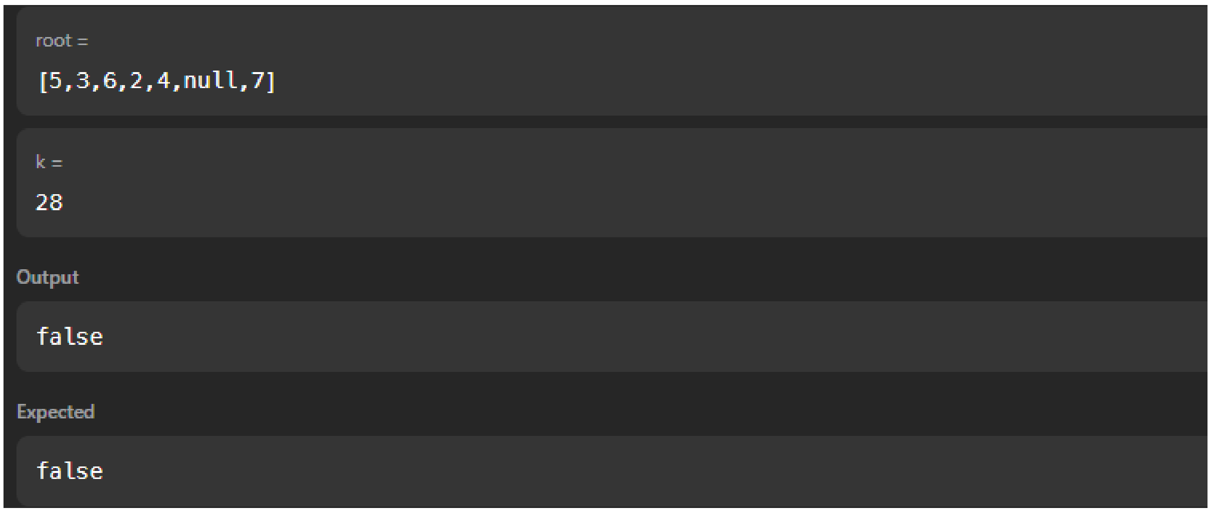
```
9
```

Output

```
true
```

Expected

```
true
```



root =

```
[5,3,6,2,4,null,7]
```

k =

```
28
```

Output

```
false
```

Expected

```
false
```