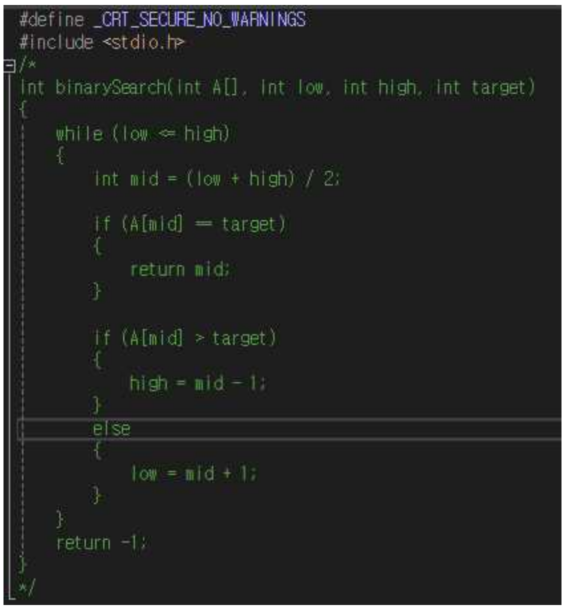
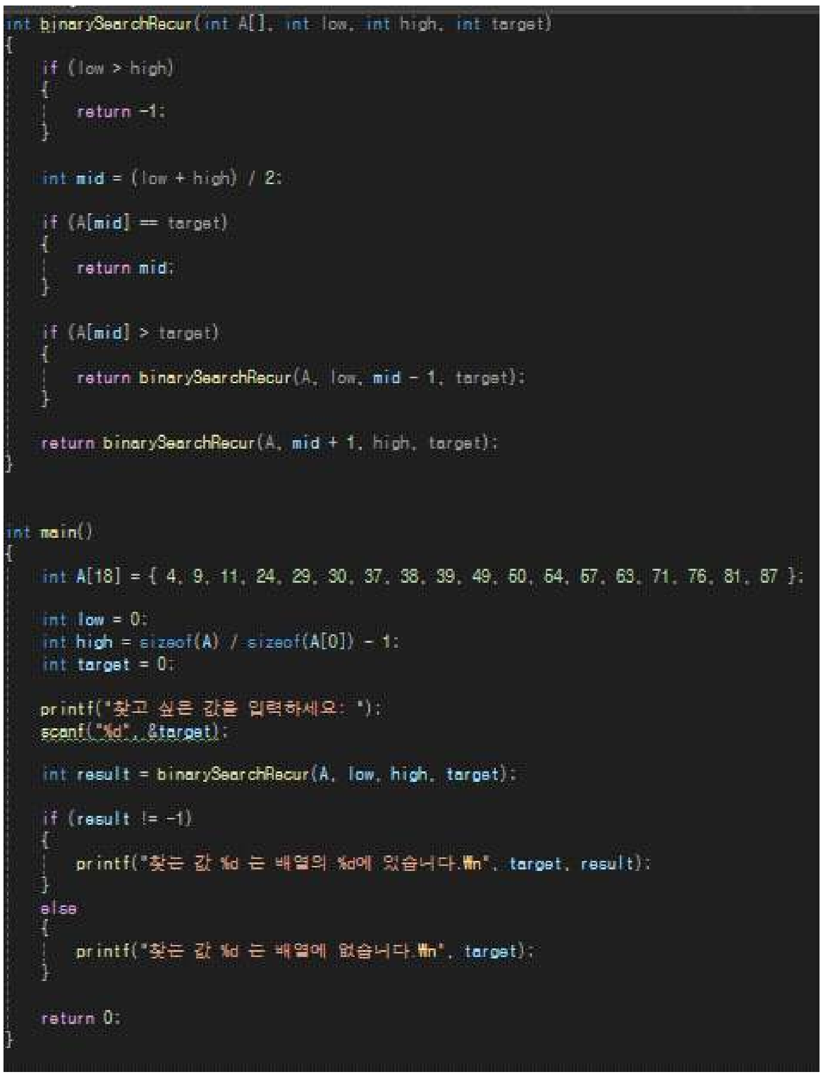
1. Binary Search Algorithm

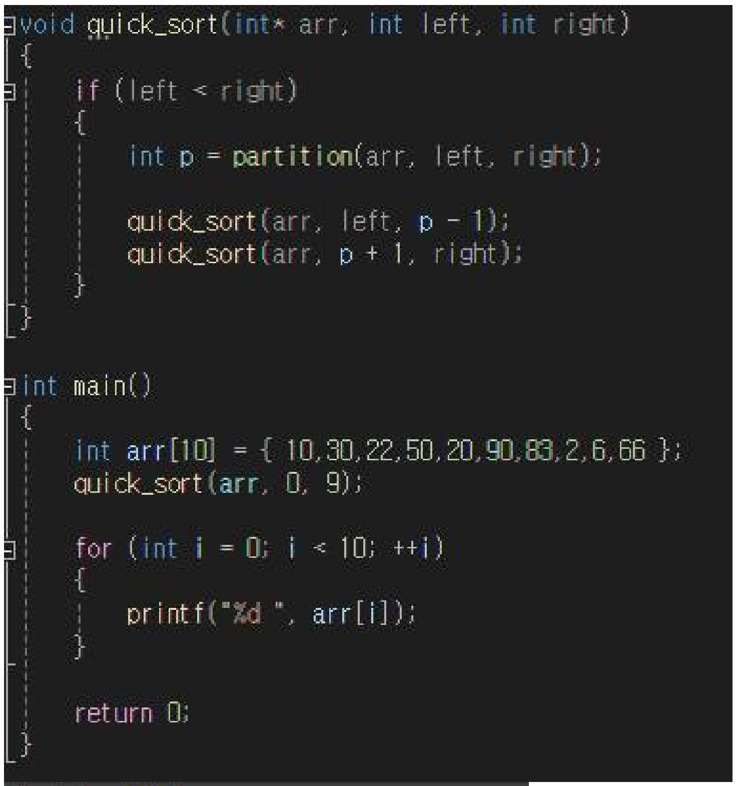


```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
/*
int binarySearch(int A[], int low, int high, int target)
{
    while (low <= high)
    {
        int mid = (low + high) / 2;

        if (A[mid] == target)
        {
            return mid;
        }

        if (A[mid] > target)
        {
            high = mid - 1;
        }
        else
        {
            low = mid + 1;
        }
    }
    return -1;
}
*/
```

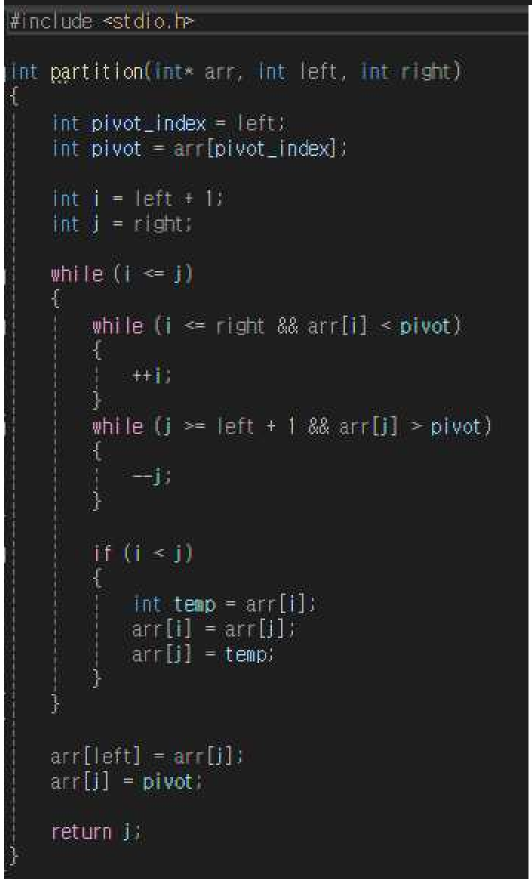


```c
int binarySearchRecur(int A[], int low, int high, int target)
{
    if (low > high)
    {
        return -1;
    }

    int mid = (low + high) / 2;

    if (A[mid] == target)
    {
        return mid;
    }

    if (A[mid] > target)
    {
        return binarySearchRecur(A, low, mid - 1, target);
    }

    return binarySearchRecur(A, mid + 1, high, target);
}


int main()
{
    int A[18] = { 4, 9, 11, 24, 29, 30, 37, 38, 39, 49, 60, 64, 67, 63, 71, 76, 81, 87 };

    int low = 0;
    int high = sizeof(A) / sizeof(A[0]) - 1;
    int target = 0;

    printf("찾고 싶은 값을 입력하세요: ");
    scanf("%d", &target);

    int result = binarySearchRecur(A, low, high, target);

    if (result != -1)
    {
        printf("찾는 값 %d 는 배열의 %d에 있습니다.₩n", target, result);
    }
    else
    {
        printf("찾는 값 %d 는 배열에 없습니다.₩n", target);
    }

    return 0;
}
```

2. Quick Sort Algorithm



```c
void quick_sort(int* arr, int left, int right)
{
    if (left < right)
    {
        int p = partition(arr, left, right);

        quick_sort(arr, left, p - 1);
        quick_sort(arr, p + 1, right);
    }
}

int main()
{
    int arr[10] = { 10,30,22,50,20,90,83,2,6,66 };
    quick_sort(arr, 0, 9);

    for (int i = 0; i < 10; ++i)
    {
        printf("%d ", arr[i]);
    }

    return 0;
}
```



```c
#include <stdio.h>

int partition(int* arr, int left, int right)
{
    int pivot_index = left;
    int pivot = arr[pivot_index];

    int i = left + 1;
    int j = right;

    while (i <= j)
    {
        while (i <= right && arr[i] < pivot)
        {
            ++i;
        }
        while (j >= left + 1 && arr[j] > pivot)
        {
            --j;
        }

        if (i < j)
        {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    arr[left] = arr[j];
    arr[j] = pivot;

    return j;
}
```