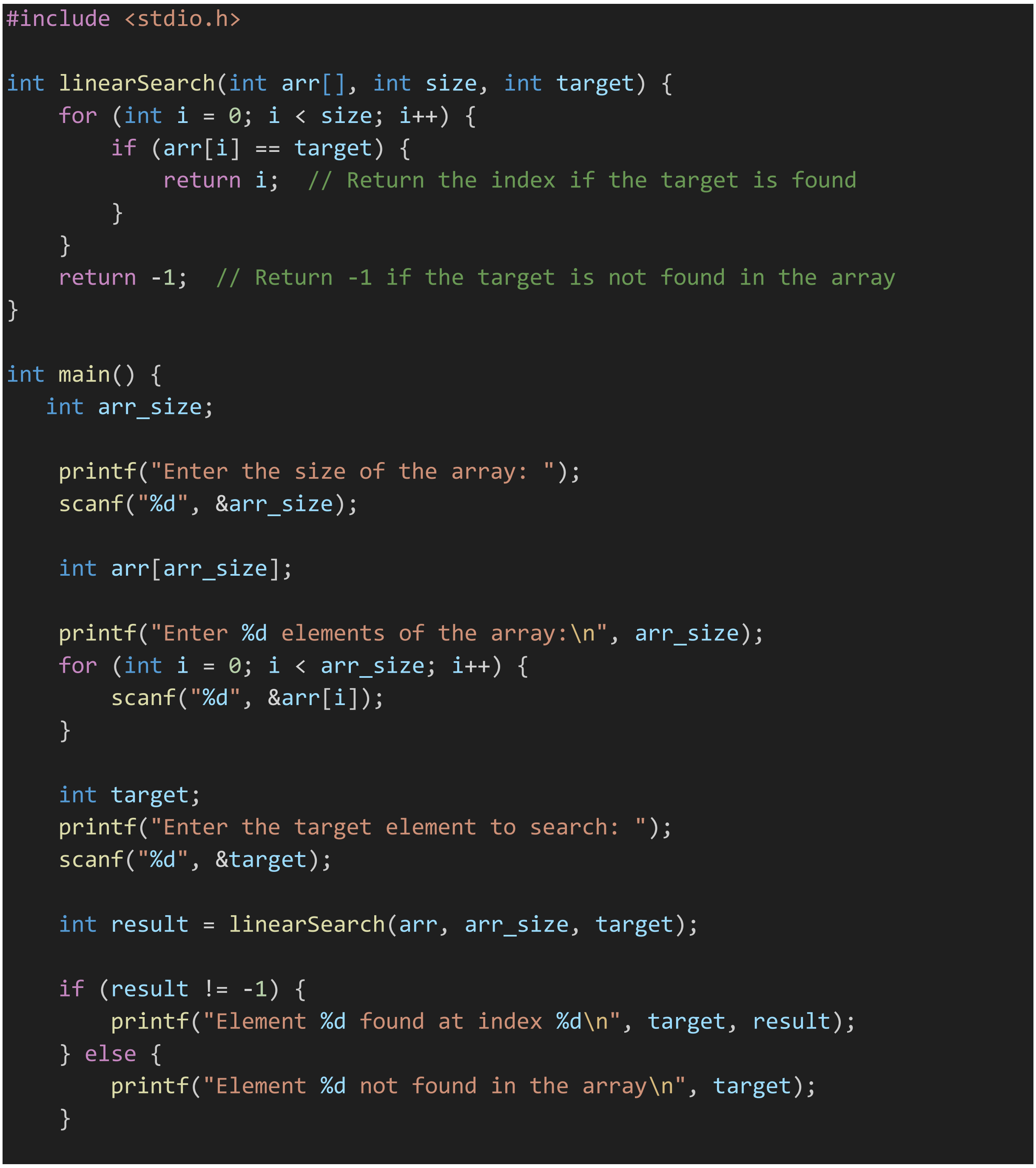
Lab 3

Mark lopes

9913

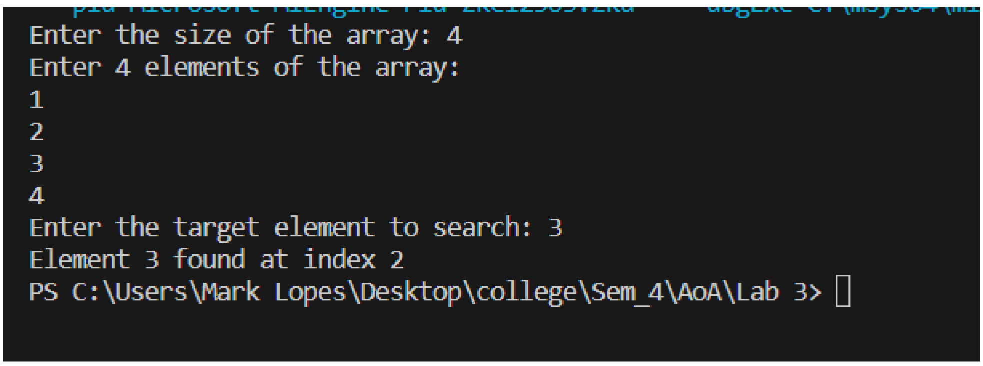
SE-Comps_A Batch-C

Linear search:-



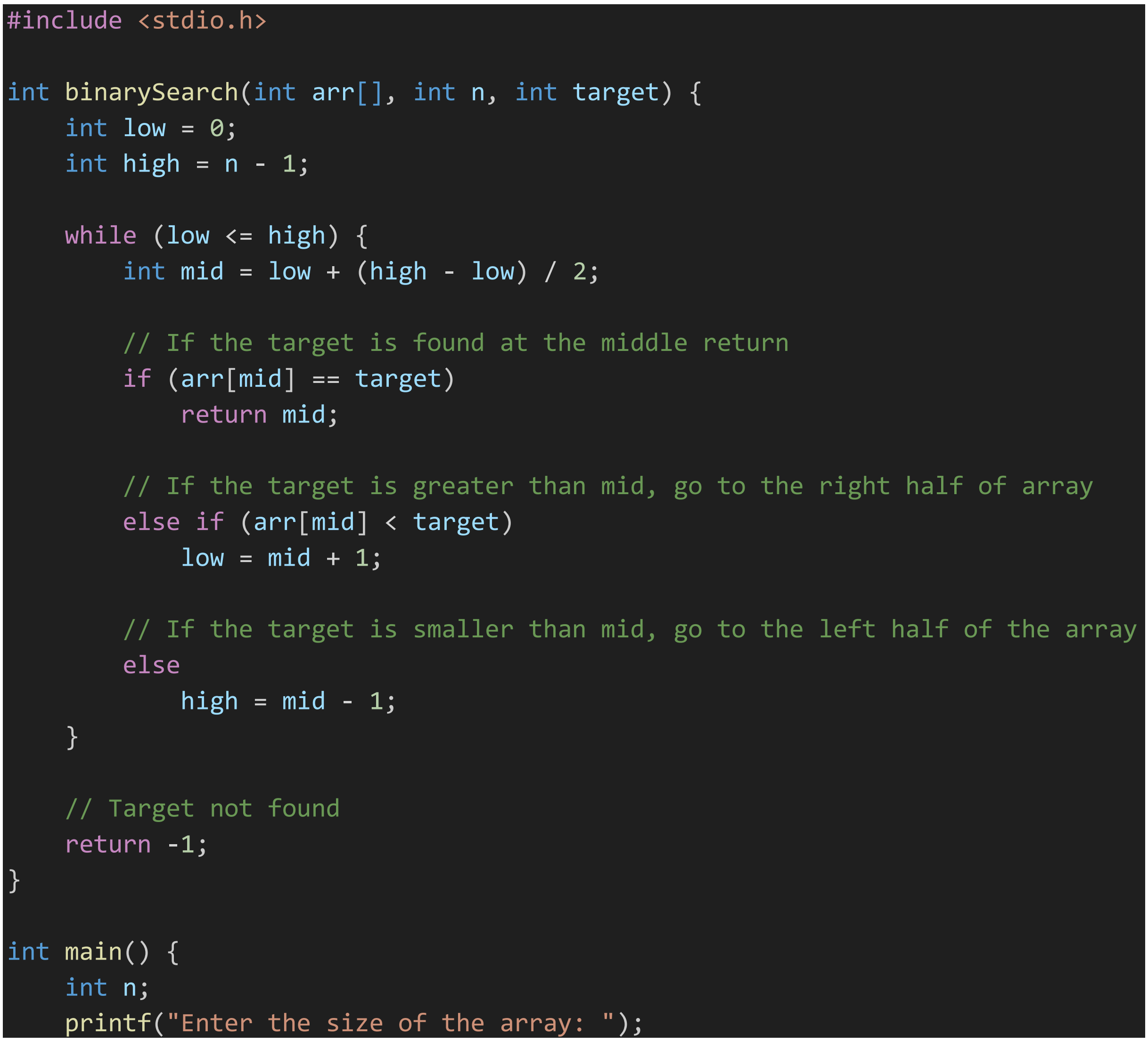
```c
#include <stdio.h>

int linearSearch(int arr[], int size, int target) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            return i;  // Return the index if the target is found
        }
    }
    return -1;  // Return -1 if the target is not found in the array
}

int main() {
   int arr_size;

    printf("Enter the size of the array: ");
    scanf("%d", &arr_size);

    int arr[arr_size];

    printf("Enter %d elements of the array:\n", arr_size);
    for (int i = 0; i < arr_size; i++) {
        scanf("%d", &arr[i]);
    }

    int target;
    printf("Enter the target element to search: ");
    scanf("%d", &target);

    int result = linearSearch(arr, arr_size, target);

    if (result != -1) {
        printf("Element %d found at index %d\n", target, result);
    } else {
        printf("Element %d not found in the array\n", target);
    }
```

```c
    return 0;
}
```

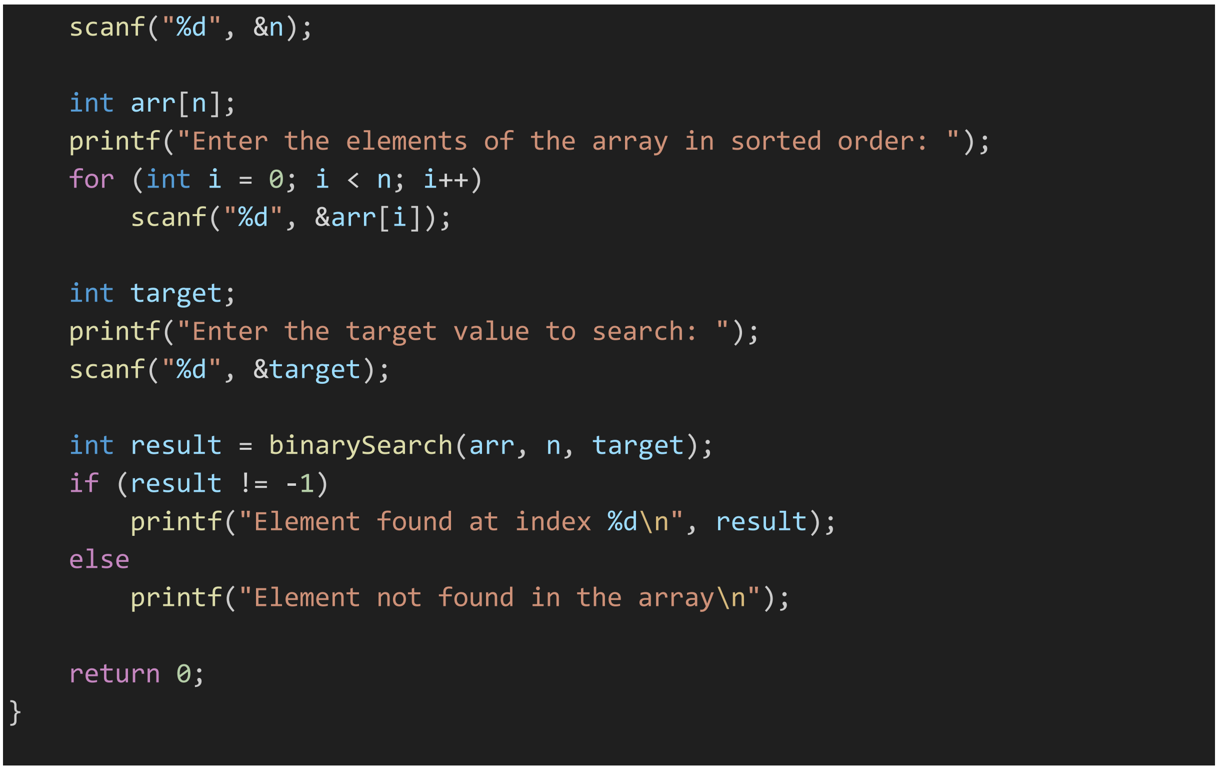


```
Enter the size of the array: 4
Enter 4 elements of the array:
1
2
3
4
Enter the target element to search: 3
Element 3 found at index 2
PS C:\Users\Mark Lopes\Desktop\college\Sem_4\AoA\Lab 3> 
```

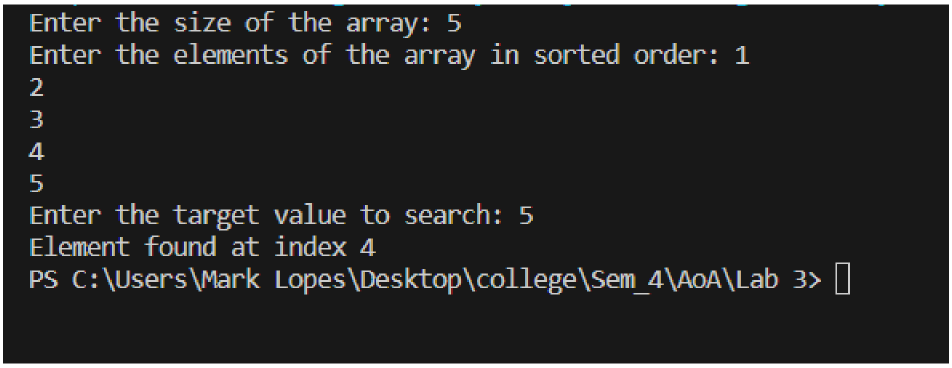
Binary search:-



```c
#include <stdio.h>

int binarySearch(int arr[], int n, int target) {
    int low = 0;
    int high = n - 1;

    while (low <= high) {
        int mid = low + (high - low) / 2;

        // If the target is found at the middle return
        if (arr[mid] == target)
            return mid;

        // If the target is greater than mid, go to the right half of array
        else if (arr[mid] < target)
            low = mid + 1;

        // If the target is smaller than mid, go to the left half of the array
        else
            high = mid - 1;
    }

    // Target not found
    return -1;
}

int main() {
    int n;
    printf("Enter the size of the array: ");
```

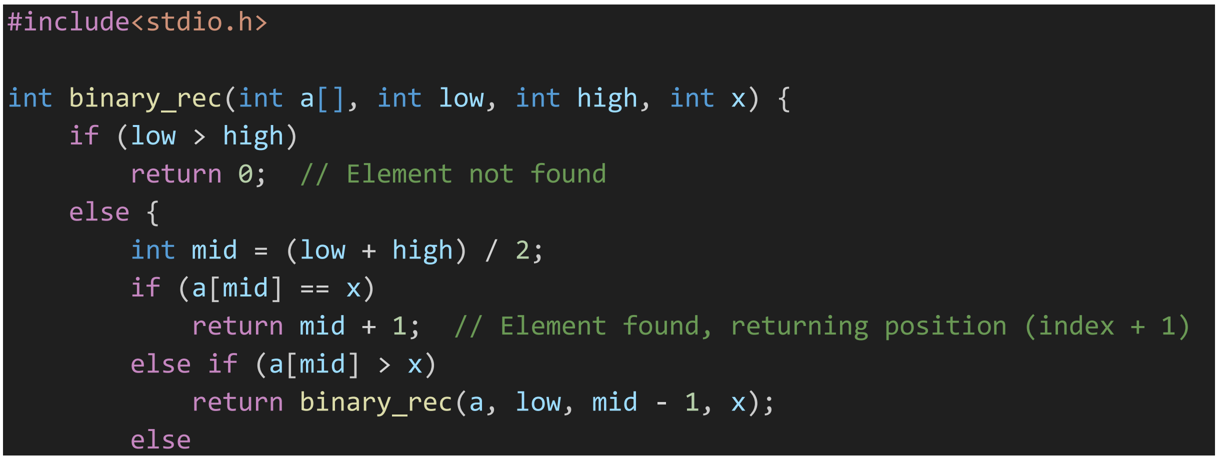


```
    scanf("%d", &n);

    int arr[n];
    printf("Enter the elements of the array in sorted order: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    int target;
    printf("Enter the target value to search: ");
    scanf("%d", &target);

    int result = binarySearch(arr, n, target);
    if (result != -1)
        printf("Element found at index %d\n", result);
    else
        printf("Element not found in the array\n");

    return 0;
}
```

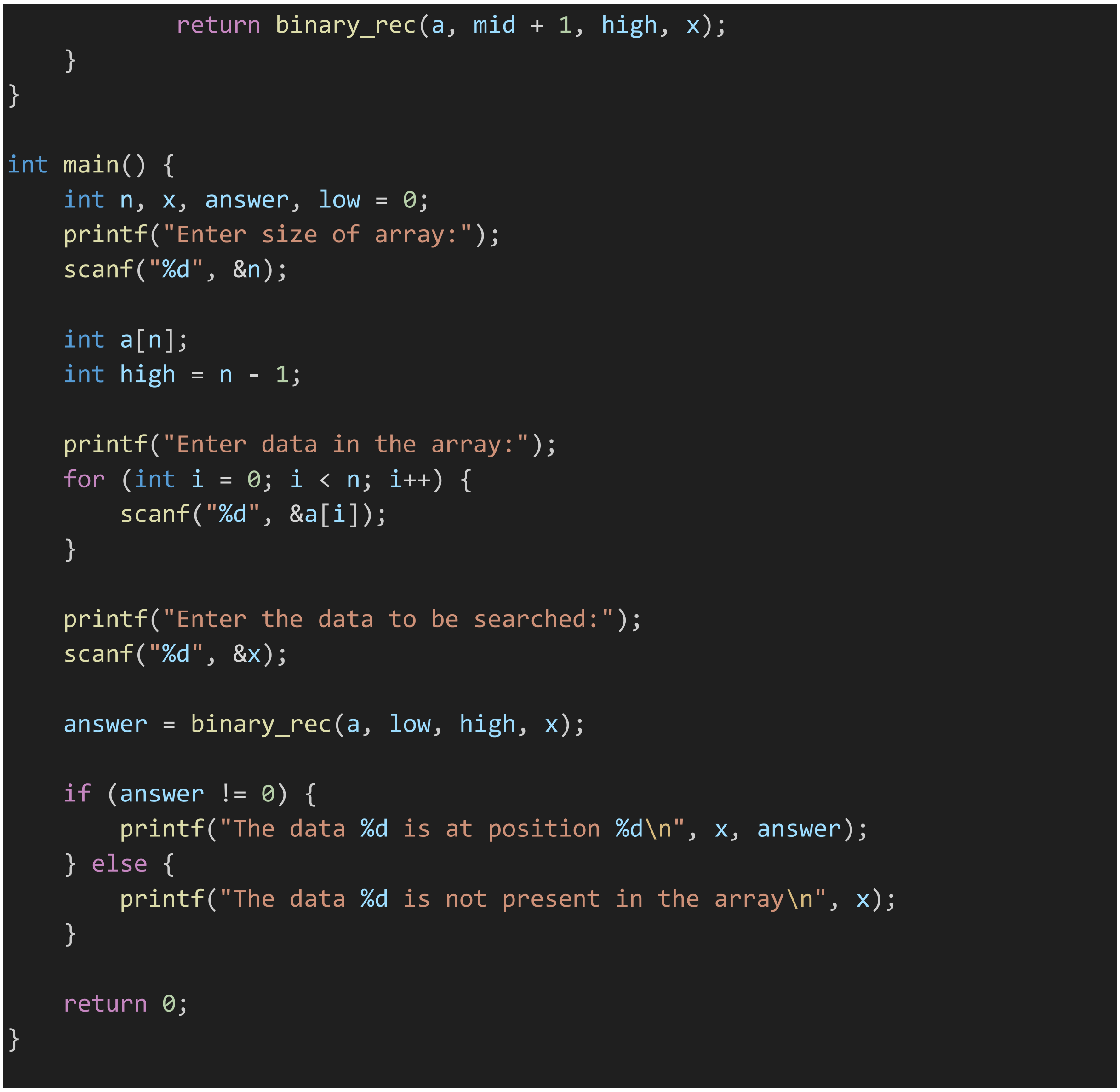


```
Enter the size of the array: 5
Enter the elements of the array in sorted order: 1
2
3
4
5
Enter the target value to search: 5
Element found at index 4
PS C:\Users\Mark Lopes\Desktop\college\Sem_4\AoA\Lab 3> 
```

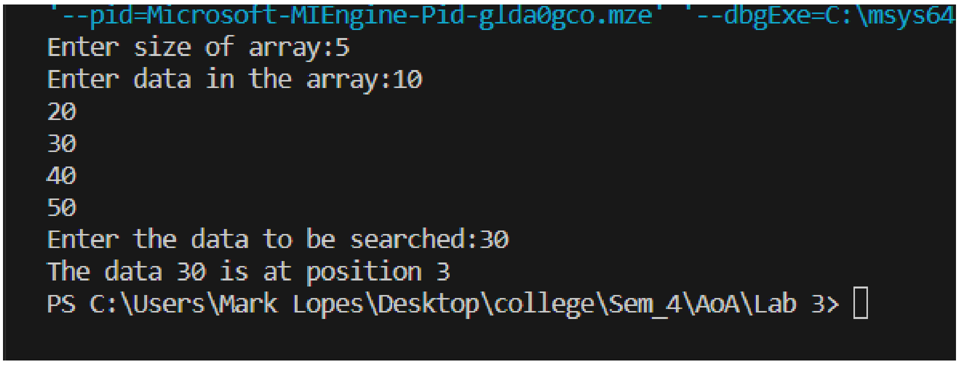
Binary recursion:-



```
#include<stdio.h>

int binary_rec(int a[], int low, int high, int x) {
    if (low > high)
        return 0;   // Element not found
    else {
        int mid = (low + high) / 2;
        if (a[mid] == x)
            return mid + 1;   // Element found, returning position (index + 1)
        else if (a[mid] > x)
            return binary_rec(a, low, mid - 1, x);
        else
```

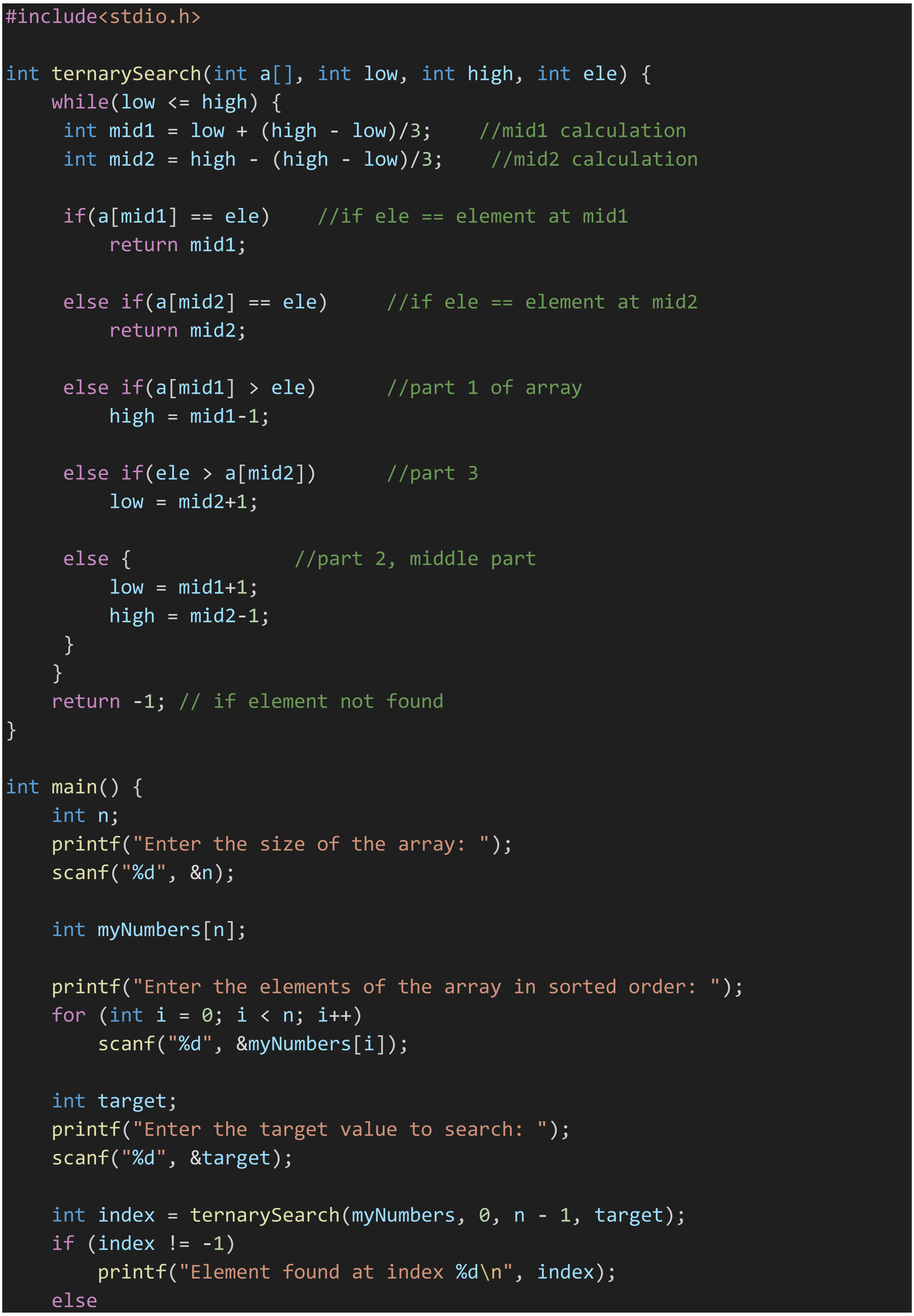


```
            return binary_rec(a, mid + 1, high, x);
    }
}

int main() {
    int n, x, answer, low = 0;
    printf("Enter size of array:");
    scanf("%d", &n);

    int a[n];
    int high = n - 1;

    printf("Enter data in the array:");
    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    printf("Enter the data to be searched:");
    scanf("%d", &x);

    answer = binary_rec(a, low, high, x);

    if (answer != 0) {
        printf("The data %d is at position %d\n", x, answer);
    } else {
        printf("The data %d is not present in the array\n", x);
    }

    return 0;
}
```

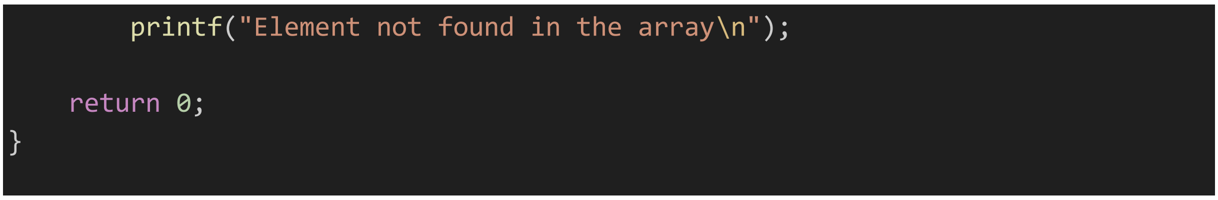


```
'--pid=Microsoft-MIEngine-Pid-glda0gco.mze' '--dbgExe=C:\msys64
Enter size of array:5
Enter data in the array:10
20
30
40
50
Enter the data to be searched:30
The data 30 is at position 3
PS C:\Users\Mark Lopes\Desktop\college\Sem_4\AoA\Lab 3> 
```

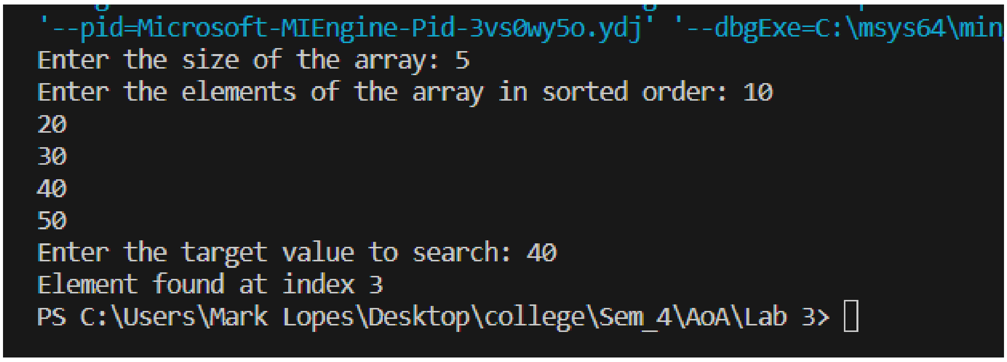
Ternary:-



```
#include<stdio.h>

int ternarySearch(int a[], int low, int high, int ele) {
    while(low <= high) {
     int mid1 = low + (high - low)/3;    //mid1 calculation
     int mid2 = high - (high - low)/3;    //mid2 calculation

     if(a[mid1] == ele)    //if ele == element at mid1
         return mid1;

     else if(a[mid2] == ele)     //if ele == element at mid2
         return mid2;

     else if(a[mid1] > ele)      //part 1 of array
         high = mid1-1;

     else if(ele > a[mid2])      //part 3
         low = mid2+1;

     else {              //part 2, middle part
         low = mid1+1;
         high = mid2-1;
     }
    }
    return -1; // if element not found
}

int main() {
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int myNumbers[n];

    printf("Enter the elements of the array in sorted order: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &myNumbers[i]);

    int target;
    printf("Enter the target value to search: ");
    scanf("%d", &target);

    int index = ternarySearch(myNumbers, 0, n - 1, target);
    if (index != -1)
        printf("Element found at index %d\n", index);
    else
```

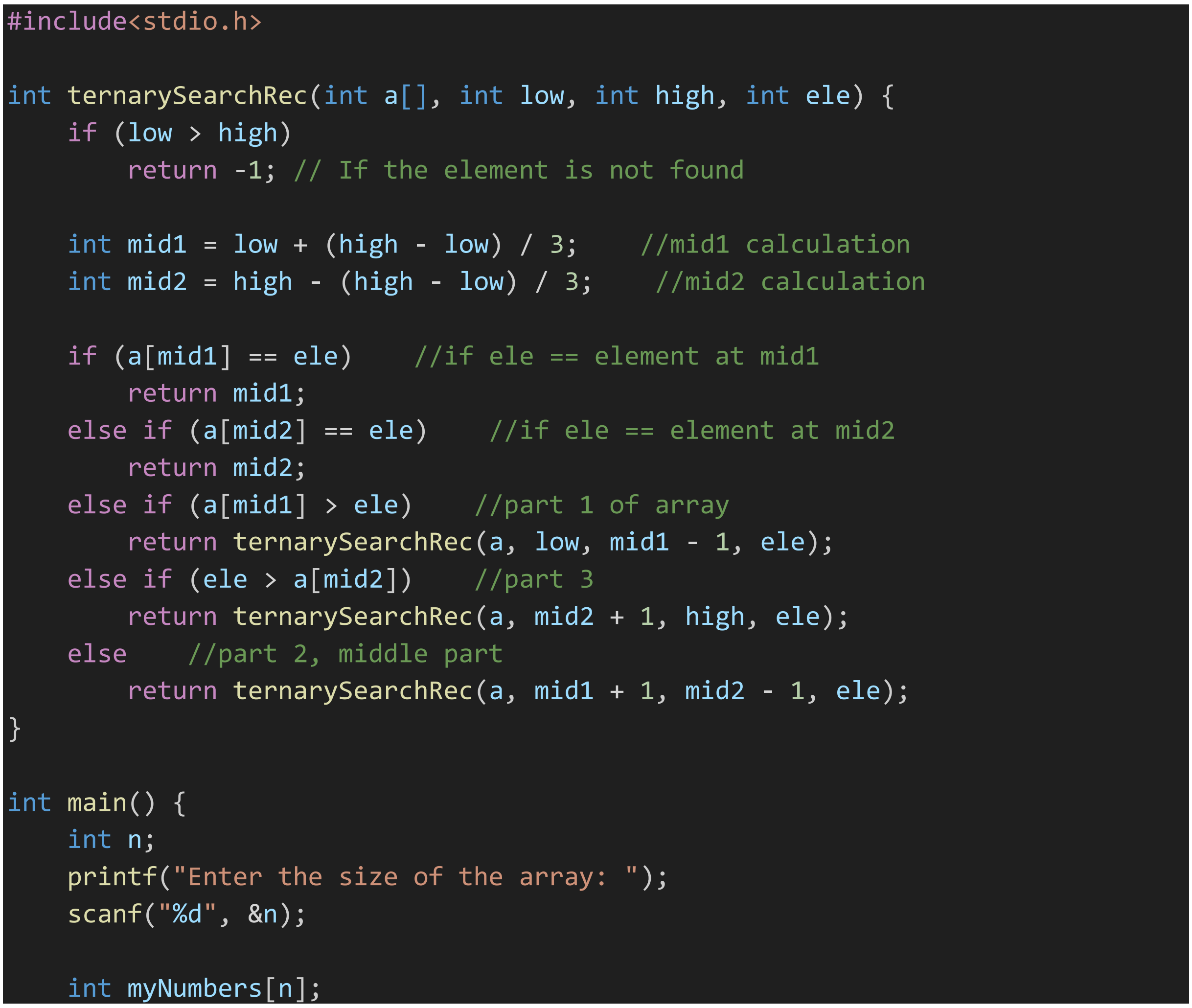


```c
        printf("Element not found in the array\n");

    return 0;
}
```

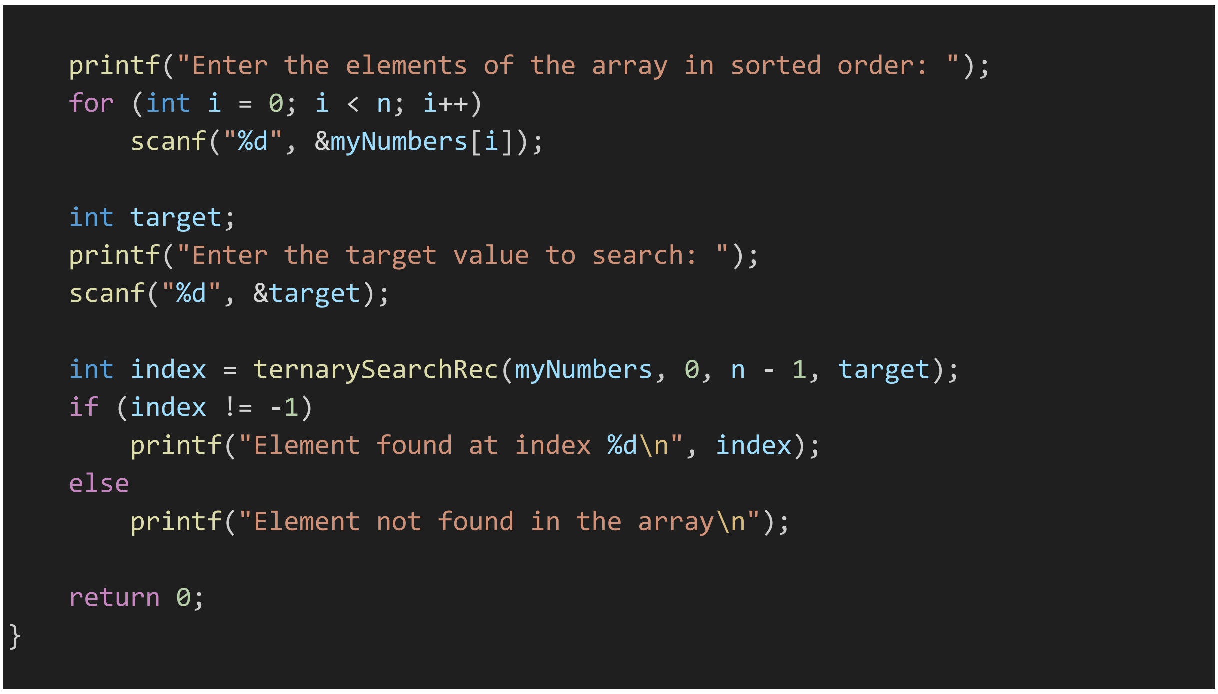


```
 '--pid=Microsoft-MIEngine-Pid-3vs0wy5o.ydj' '--dbgExe=C:\msys64\min
 Enter the size of the array: 5
 Enter the elements of the array in sorted order: 10
 20
 30
 40
 50
 Enter the target value to search: 40
 Element found at index 3
 PS C:\Users\Mark Lopes\Desktop\college\Sem_4\AoA\Lab 3> 
```

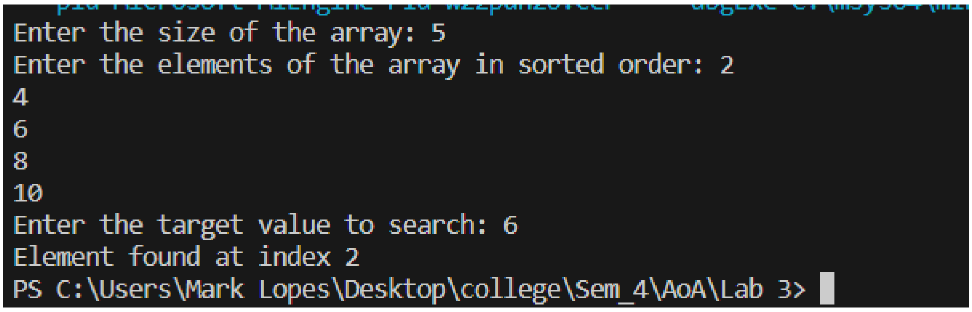
Ternary recursion:-



```c
#include<stdio.h>

int ternarySearchRec(int a[], int low, int high, int ele) {
    if (low > high)
        return -1; // If the element is not found

    int mid1 = low + (high - low) / 3;    //mid1 calculation
    int mid2 = high - (high - low) / 3;    //mid2 calculation

    if (a[mid1] == ele)    //if ele == element at mid1
        return mid1;
    else if (a[mid2] == ele)    //if ele == element at mid2
        return mid2;
    else if (a[mid1] > ele)    //part 1 of array
        return ternarySearchRec(a, low, mid1 - 1, ele);
    else if (ele > a[mid2])    //part 3
        return ternarySearchRec(a, mid2 + 1, high, ele);
    else    //part 2, middle part
        return ternarySearchRec(a, mid1 + 1, mid2 - 1, ele);
}

int main() {
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int myNumbers[n];
```

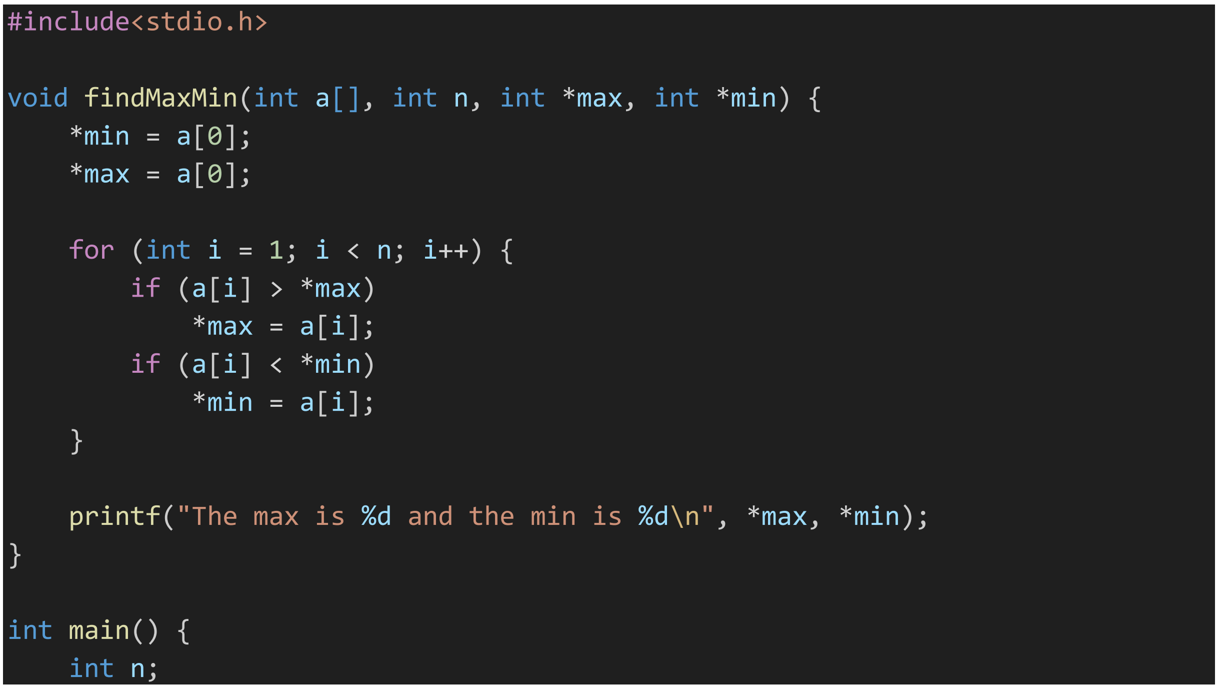


```
    printf("Enter the elements of the array in sorted order: ");
    for (int i = 0; i < n; i++)
        scanf("%d", &myNumbers[i]);

    int target;
    printf("Enter the target value to search: ");
    scanf("%d", &target);

    int index = ternarySearchRec(myNumbers, 0, n - 1, target);
    if (index != -1)
        printf("Element found at index %d\n", index);
    else
        printf("Element not found in the array\n");

    return 0;
}
```

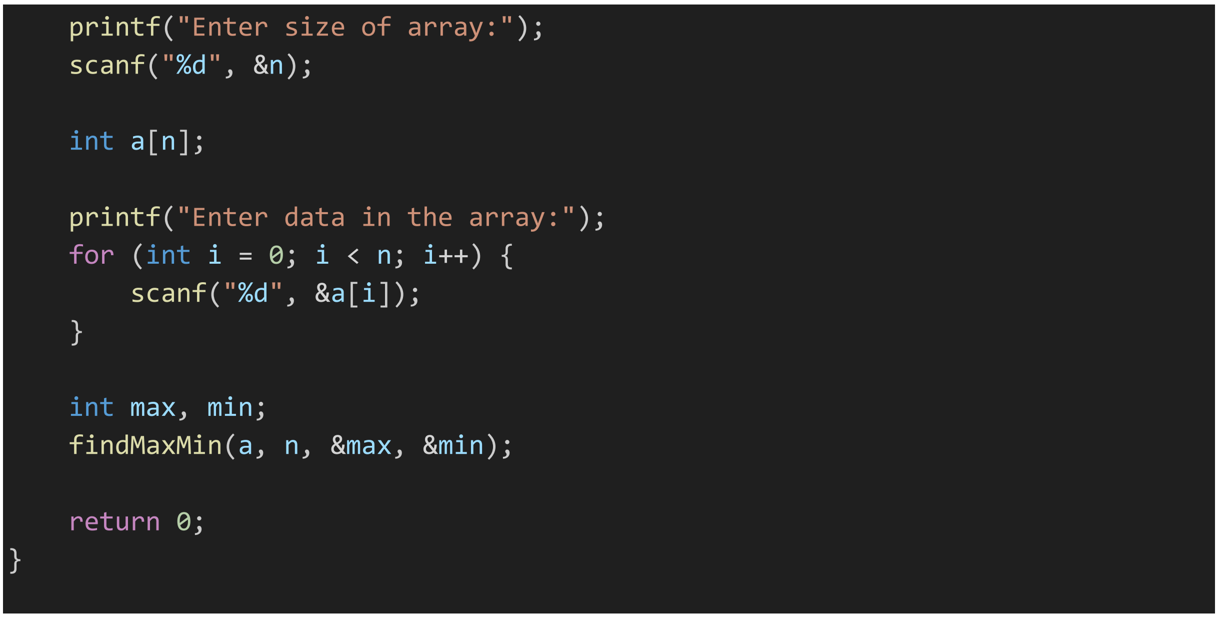


```
Enter the size of the array: 5
Enter the elements of the array in sorted order: 2
4
6
8
10
Enter the target value to search: 6
Element found at index 2
PS C:\Users\Mark Lopes\Desktop\college\Sem_4\AoA\Lab 3> 
```

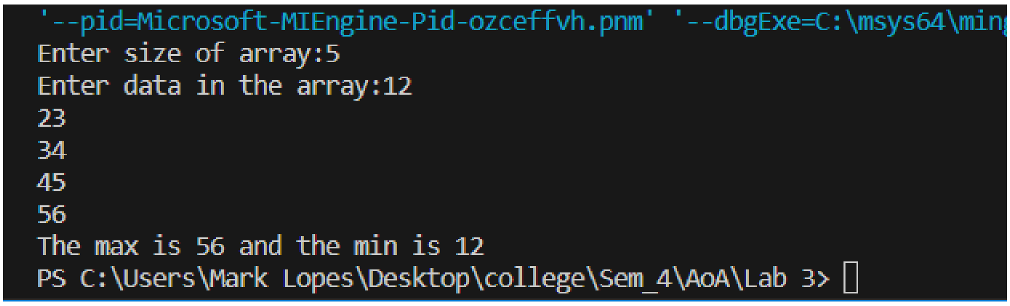
MinMax Brute force:-



```
#include<stdio.h>

void findMaxMin(int a[], int n, int *max, int *min) {
    *min = a[0];
    *max = a[0];

    for (int i = 1; i < n; i++) {
        if (a[i] > *max)
            *max = a[i];
        if (a[i] < *min)
            *min = a[i];
    }

    printf("The max is %d and the min is %d\n", *max, *min);
}

int main() {
    int n;
```

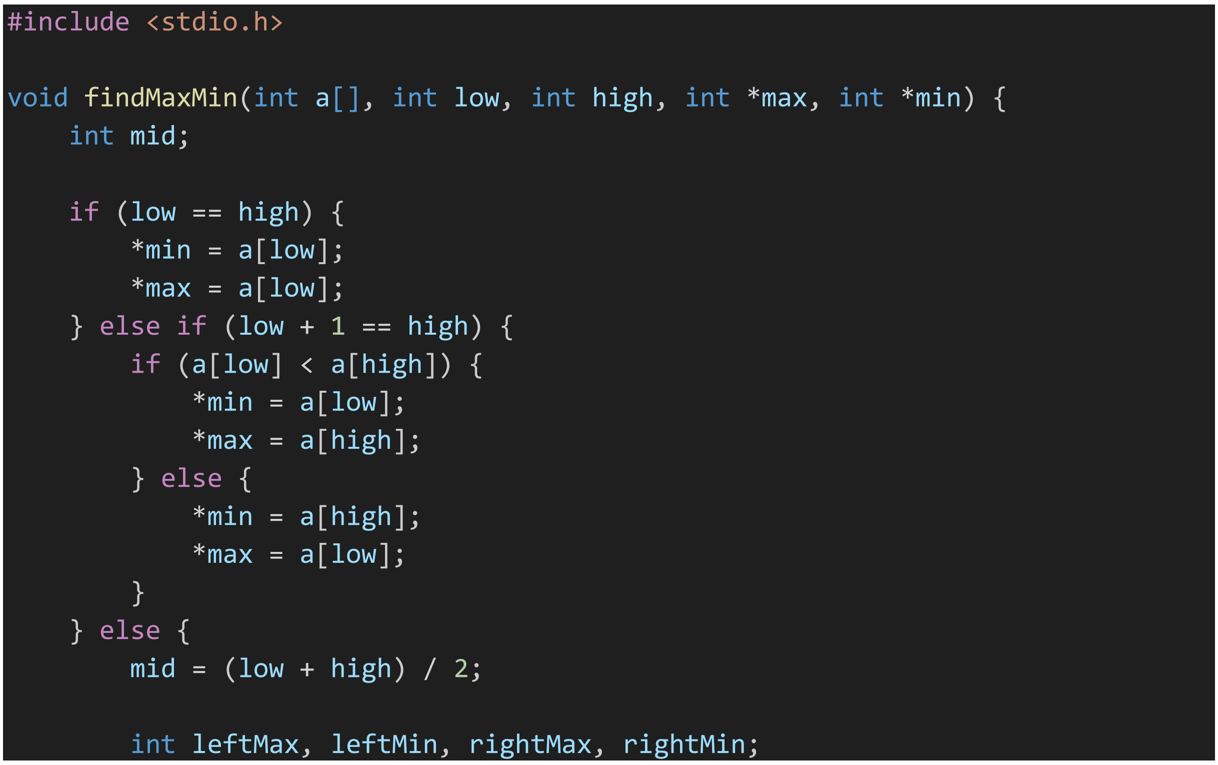


```
    printf("Enter size of array:");
    scanf("%d", &n);

    int a[n];

    printf("Enter data in the array:");
    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    int max, min;
    findMaxMin(a, n, &max, &min);

    return 0;
}
```

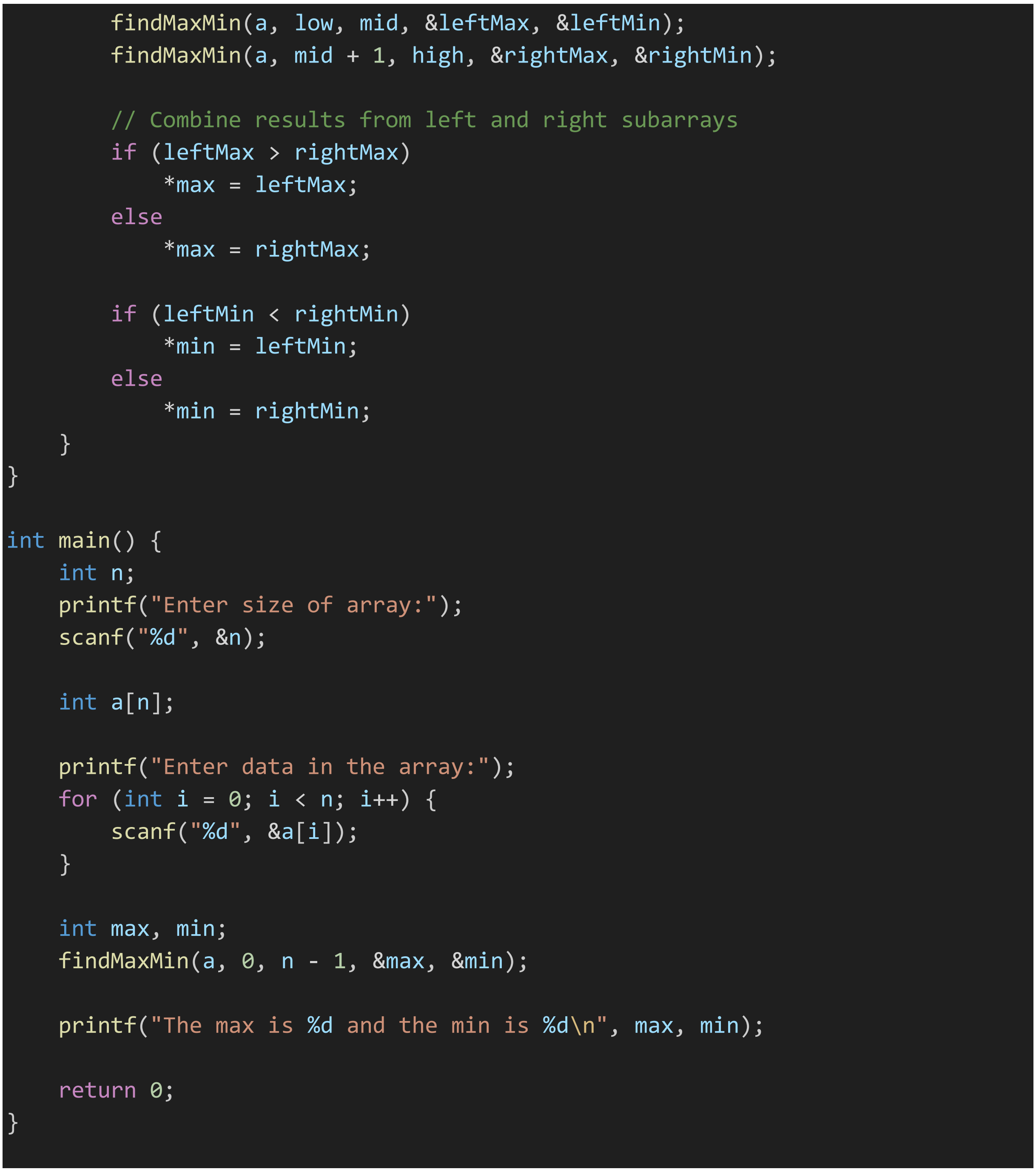


```
 '--pid=Microsoft-MIEngine-Pid-ozceffvh.pnm' '--dbgExe=C:\msys64\ming
 Enter size of array:5
 Enter data in the array:12
 23
 34
 45
 56
 The max is 56 and the min is 12
 PS C:\Users\Mark Lopes\Desktop\college\Sem_4\AoA\Lab 3> 
```

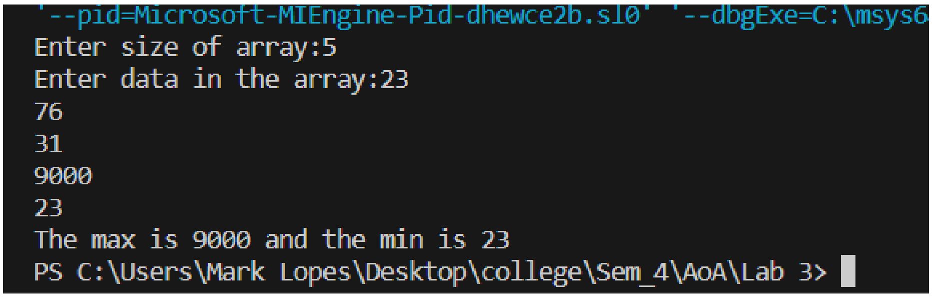
MinMax recursion:-



```
#include <stdio.h>

void findMaxMin(int a[], int low, int high, int *max, int *min) {
    int mid;

    if (low == high) {
        *min = a[low];
        *max = a[low];
    } else if (low + 1 == high) {
        if (a[low] < a[high]) {
            *min = a[low];
            *max = a[high];
        } else {
            *min = a[high];
            *max = a[low];
        }
    } else {
        mid = (low + high) / 2;

        int leftMax, leftMin, rightMax, rightMin;
```



```
        findMaxMin(a, low, mid, &leftMax, &leftMin);
        findMaxMin(a, mid + 1, high, &rightMax, &rightMin);

        // Combine results from left and right subarrays
        if (leftMax > rightMax)
            *max = leftMax;
        else
            *max = rightMax;

        if (leftMin < rightMin)
            *min = leftMin;
        else
            *min = rightMin;
    }
}

int main() {
    int n;
    printf("Enter size of array:");
    scanf("%d", &n);

    int a[n];

    printf("Enter data in the array:");
    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    int max, min;
    findMaxMin(a, 0, n - 1, &max, &min);

    printf("The max is %d and the min is %d\n", max, min);

    return 0;
}
```



```
 '--pid=Microsoft-MIEngine-Pid-dhewce2b.sl0' '--dbgExe=C:\msys6
 Enter size of array:5
 Enter data in the array:23
 76
 31
 9000
 23
 The max is 9000 and the min is 23
 PS C:\Users\Mark Lopes\Desktop\college\Sem_4\AoA\Lab 3> █
```

~~Lab~~ 3.

Lab 3 Postlab

Q. Comment on complexity of each of the search technique.

→

1. Linear Search

i) Time complexity = O(n)

ii) The algorithm checks every element ~~and~~ until the target is found or end is reached. ∴ complexity in worst case is size(n).

2. Binary search

i) Time complexity is O(log n)

ii) It is a divide and conquer algorithm that works only on a sorted array. Because the array is halved in each step the complexity is O(log n).

3 Ternary search

i) Time complexity = O(log n)

ii) Ternary search is also a divide and conquer but it divides the array into three parts instead of two ∴ the complexity is still logrithmic

Q.2

→ ~~Terna~~ Ternary search is not that much better because, even though it has a slightly high base algorithm ($\log_3 n$) compa to binary ($\log_2 n$), it does not offer a signifi improvement in terms of time complexity. In practice, the difference in performance is almost negligible especially for small datas. Binary search is more commonly used due to it being simple.

Q.3 Discuss fake coin problem in detail.

→ i) Fake coin problem is a problem where we find a fake coin out of a number of coins.
ii) The fake coin is assumed lighter than the real coins using a balance scale.

There are Two ways of solving it :-

1] Brute force method :-

1. In this method, we pick a coin at random and use it as a test coin for other remaining ones.
2. We will test it one by one with all the coins, and if the ~~tst~~ balance towards a side we can figure out the fake coin.

Algorithm:-

1. First coin from top will be chosen as test coin
2. First we compare the first and second coin. If they are ~~ba~~ balanced, then we will proceed to compare ~~for~~ first and third coin, and then first and fourth and so on.
3. If we get an imbalance, the coin with which we get our imbalance is fake coin.

As we compare with all elements, time complexity is O(n).

2] Decrease and Conquer technique

Algorithm:-

1. Take number of coins = n
2. If there is only 1 coin, then that is fake coin.
3. if there are more than one coin, then divide it into piles of A = ceiling (n/3), B = ceiling (n/3), C = n - 2* ceiling (n/3).
4. Weigh A and B, if scale matches then repeat from first step with total number of coins equalling number of coins in C.
5. If scale is unbalanced, then repeat from step 1, with the lighter of A and B.

As we divide the problem, time complexity is O(log n).