

5-Implementation of Quick Sort

Started on: Tuesday, 9 September 2023, 12:24 PM

Status: Finished

Completed on: Tuesday, 9 September 2023, 12:25 PM

Time taken: 45 sec

Marks: 1.00/1.00

Grade: 10.00 out of 10.00 (100%)

Question 1: Score: 1.00/1.00 out of 1.00 [View question](#)

Write a Program to Implement the Quick Sort Algorithm.

Input Format:

The first line contains the no. of elements in the list.

The next n lines contain the elements.

Output:

Sorted list of elements.

For example:

Input	Result
9	12 34 47 78 99
67 34 12 99 78	

Answer:

```
1. #include <iostream.h>
2.
3. void swap(int *a, int *b) {
4.     int t = *a; *a = *b; *b = t;
5. }
6.
7. int partition(int arr[], int low, int high) {
8.     int pivot = arr[high], i = low - 1;
9.     for (int j = low; j < high; j++) {
10.         if (arr[j] < pivot) swap(&arr[i], &arr[j]);
11.         i++;
12.     }
13.     swap(&arr[i + 1], &arr[high]);
14.     return i + 1;
15. }
16.
17. void quickSort(int arr[], int low, int high) {
18.     if (low <= high) {
19.         int pi = partition(arr, low, high);
20.         quickSort(arr, low, pi - 1);
21.         quickSort(arr, pi + 1, high);
22.     }
23. }
24.
25. int main() {
26.     int N;
27.     cout << "N: ";
28.     cin >> N;
29.     for (int i = 0; i < N; i++) cout << arr[i] << " ";
30.     quickSort(arr, 0, N - 1);
31.     for (int i = 0; i < N; i++) cout << arr[i] << " ";
32.     return 0;
33. }
```

```
2. void swap(int *a, int *b) {
3.     int t = *a; *a = *b; *b = t;
4. }
5.
6. int partition(int arr[], int low, int high) {
7.     int pivot = arr[high], i = low - 1;
8.     for (int j = low; j < high; j++) {
9.         if (arr[j] < pivot) swap(&arr[i], &arr[j]);
10.        i++;
11.    }
12.    swap(&arr[i + 1], &arr[high]);
13.    return i + 1;
14. }
15.
16. void quickSort(int arr[], int low, int high) {
17.     if (low <= high) {
18.         int pi = partition(arr, low, high);
19.         quickSort(arr, low, pi - 1);
20.         quickSort(arr, pi + 1, high);
21.     }
22. }
23.
24. int main() {
25.     int N;
26.     cout << "N: ";
27.     cin >> N;
28.     for (int i = 0; i < N; i++) cout << arr[i] << " ";
29.     quickSort(arr, 0, N - 1);
30.     for (int i = 0; i < N; i++) cout << arr[i] << " ";
31.     return 0;
32. }
```

Input	Expected	Get
#	12 34 47 78 99	12 34 47 78 99
67 34 12 99 78		
#	1 18 11 22 18 10 78 99 88 134 3 14 13 32 56 38 78 99 124	1 18 11 22 18 10 78 99 88 134 3 14 13 32 56 38 78 99 124
#	1 18 11 22 18 10 78 99 88 134	1 18 11 22 18 10 78 99 88 134
#	1 2 3 4 5 6 7 8 9 10 11 12 13	1 2 3 4 5 6 7 8 9 10 11 12 13
#	9 8 7 6 5 4 3 2 1 10 11 12 13	9 8 7 6 5 4 3 2 1 10 11 12 13

Passed all tests! #

Submit

Score: 1.00/1.00

4-Two Elements sum to x

Start time - Tuesday, 9 September 2025, 12:22 PM

Status - Pending

Completed on - Tuesday, 9 September 2025, 12:24 PM

Time taken - 1 min 44 sec.

Marks - 1.00/1.00

Grade - 10.00 out of 10.00 (100%)

Question 1 : Correct Marks 1.00 out of 1.00 [100%]

Problem Statement:

Given a sorted array of integers say arr and a number x. Write a recursive program using divide and conquer strategy to check if there exist two elements in the array whose sum = x; if there exist such two elements then return the numbers, otherwise print as "No".

Note: Write a Divide and Conquer Solution.

Input Format:

First Line Contains Integer n - Size of array
Next n lines Contains n numbers - Elements of an array
Last Line Contains Integer x - Sum Value

Output Format:

First Line Contains Integer - Element 1
Second Line Contains Integer - Element 2 (Element 1 and Element 2 together sums to value 'x')

Answer: (Priority regime: 0 %)

```
1. #include <iostream.h>
2. #include <conio.h>
3.
4. int search {
5.     int n, x;
6.     scanf("%d", &n);
7.     int arr[n];
8.     for (int i = 0; i < n; i++) scanf("%d", &arr[i]);
9.     scanf("%d", &x);
10.    int i = 0, j = n - 1;
11.
```

```
12.    while (i < j) {
13.        int sum = arr[i] + arr[j];
14.        if (sum == x) {
15.            printf("Element1=%d, arr[%d], arr[%d]", sum, arr[i], arr[j]);
16.            free(arr);
17.            return 0;
18.        } else if (sum < x) {
19.            i++;
20.        } else {
21.            j--;
22.        }
23.    }
24.    printf("No\n");
25.    free(arr);
26.    return 0;
27. }
```

Input	Expected	Get
4	4	4
2	10	10
4		
2		
10		
14		
✓ 5	5	5
5		
4		
6		
8		
10		
14		

Passed all testcases ✓

Correct

Marks for this submission: 1.00/1.00

```
2 int findFloor(int arr[], int low, int high, int x) {
3     int floor = -1;
4     while (low <= high) {
5         int mid = low + (high - low) / 2;
6         if (arr[mid] == x) {
7             return arr[mid];
8         } else if (arr[mid] < x) {
9             floor = arr[mid];
10            low = mid + 1;
11        } else {
12            high = mid - 1;
13        }
14    }
15    return floor;
16 }
17 int main() {
18     int n;
19     printf("n: ");
20     scanf("%d", &n);
21     int arr[10];
22     for (int i = 0; i < n; i++) {
23         scanf("%d", &arr[i]);
24     }
25     printf("\n");
26     int floorValue = findFloor(arr, 0, n - 1, 6);
27     printf("Floor: %d\n", floorValue);
28     return 0;
29 }
```

Input	Expected	Got
6	2	2 ✓
2		
2		
4		
18		
12		
22		
22		
35		
100		
120		
100		

Input	Expected	Got
6	2	2 ✓
2		
2		
4		
18		
12		
22		
22		
35		
100		
120		
100		

Passed all tests! ✓

Cancel

Memory for this submission: 1.000 MB

3-Finding Floor Value

Started on: Tuesday, 9 September 2025, 12:20 PM

Status: Pending

Completed on: Tuesday, 9 September 2025, 12:21 PM

Time taken: 1 min 43 sec

Mark: 1.00/1.00

Grade: 10.00 out of 10.00 (100%)

Question 1: Optimal - Mark 1.00 out of 1.00

Problem Statement:

Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide and conquer algorithm to find floor of x.

Input Format:

First Line Contains Integer n – Size of array

Next n Lines Contains n numbers – Elements of an array

Last Line Contains Integer x – Value for x

Output Format:

First Line Contains Integer – Floor value for x

Answer: (Verify regime: 0.4)

```
1. #include <stdio.h>
2. int FindFloor(int arr[], int low, int high, int x) {
3.     int floor = -1;
4.     while (low <= high) {
5.         int mid = low + (high - low) / 2;
6.         if (arr[mid] == x) {
7.             return arr[mid];
8.         } else if (arr[mid] < x) {
9.             low = mid + 1;
10.        } else {
11.            high = mid - 1;
12.        }
13.    }
14.    return floor;
15. }
```

```
2. int FindFloor(int arr[], int low, int high, int x) {
3.     int floor = -1;
4.     while (low <= high) {
5.         int mid = low + (high - low) / 2;
6.         if (arr[mid] == x) {
7.             floor = arr[mid];
8.             break;
9.         } else if (arr[mid] < x) {
10.            low = mid + 1;
11.        } else {
12.            high = mid - 1;
13.        }
14.    }
15.    return floor;
16. }
17. int main() {
18.     int n;
19.     scanf("%d", &n);
20.     int arr[n];
21.     for (int i = 0; i < n; i++) {
22.         scanf("%d", &arr[i]);
23.     }
24.     int x;
25.     scanf("%d", &x);
26.     int floorValue = FindFloor(arr, 0, n - 1, x);
27.     printf("%d", floorValue);
28. }
```

4 x 4
7 2
3 3 1 1 1 2 2

Answer: (empty string) 0.0

```
1. #include <stdio.h>
2.
3. int majorityElement(int* nums, int numSize) {
4.     int majorityCount = numSize / 2;
5.     for (int i = 0; i < numSize; i++) {
6.         int count = 0;
7.         for (int j = 0; j < numSize; j++) {
8.             if (nums[j] == nums[i]) {
9.                 count++;
10.            }
11.        }
12.        if (count > majorityCount) {
13.            return nums[i];
14.        }
15.    }
16.    return -1;
17. }
18.
19. int main() {
20.     int n;
21.     scanf("%d", &n);
22.
23.     int nums[n];
24.     for (int i = 0; i < n; i++) {
25.         scanf("%d", &nums[i]);
26.     }
27.     int result = majorityElement(nums, n);
28.     printf("\n%d", result);
29.     return 0;
30. }
```

Input	Expected	Get
v. 3 2 0 -v. 1 2 3	v.	v.

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00

3-Finding Floor Value

Started on: Tuesday, 9 September 2023, 12:20 PM

Status: Finished

Completed on: Tuesday, 9 September 2023, 12:21 PM

Time taken: 1 min 45 secs

Marks: 1.00/1.00

2-Majority Element

Started on: Tuesday, 9 September 2025, 12:14 PM

Status: Finished

Completed on: Tuesday, 9 September 2025, 12:19 PM

Time taken: 5 mins 59 secs

Marks: 1.00/1.00

Grade: 10.00 out of 10.00 (100%)

Question 1: [Solve](#) [Mark](#) [Solve again](#) [Report issue](#)

Given an array nums of size n, return the majority element.

The majority element is the element that appears more than $(\lceil \frac{n}{2} \rceil)$ times. You may assume that the majority element always exists in the array.

Example 1:

Input: nums = [3,2,3]

Output: 3

Example 2:

Input: nums = [2,2,1,1,1,2,2]

Output: 2

Constraints:

- $n == \text{nums.length}$
- $1 \leq n \leq 5 \times 10^4$
- $-2^{31} \leq \text{nums}[i] \leq 2^{31} - 1$

For example:

[Input](#)

[Result](#)

```
3 2  
3 2 3 2 3 2 3
```

Answer: (penalty regime: 0%)

```
1 //include <iostream>  
2  
3 int majorityElement(int* nums, int n) {  
4     int majorityCount = n / 2;  
5     for (int i = 0; i < n; i++) {  
6         int count = 0;  
7         for (int j = 0; j < n; j++) {  
8             if (nums[i] == nums[j]) {  
9                 count++;  
10            }  
11        }  
12        if (count > majorityCount) {  
13            return nums[i];  
14        }  
15    }  
16    return -1;  
17}  
18  
19 int main() {  
20     int n;  
21     scanf("%d", &n);  
22  
23     int nums[n];  
24     for (int i = 0; i < n; i++) {  
25         scanf("%d", &nums[i]);  
26     }  
27     int result = majorityElement(nums, n);  
28     printf("%d", result);  
29 }
```



The screenshot shows a table of test results. The columns represent different test cases or variables. The rows show the expected value (left) and the actual value (right). Green checkmarks indicate successful passes, while red X's indicate failures.

	1	2	3	4	5
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	1
8	1	1	1	1	1
9	1	1	1	1	1
10	1	1	1	1	1
11	1	1	1	1	1
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	1	1	1	1
16	1	1	1	1	1
17	1	1	1	1	1
18	1	1	1	1	1
19	1	1	1	1	1
20	1	1	1	1	1
21	1	1	1	1	1
22	1	1	1	1	1
23	1	1	1	1	1
24	1	1	1	1	1
25	1	1	1	1	1
26	1	1	1	1	1
27	1	1	1	1	1
28	1	1	1	1	1
29	1	1	1	1	1
30	1	1	1	1	1
31	1	1	1	1	1
32	1	1	1	1	1
33	1	1	1	1	1
34	1	1	1	1	1
35	1	1	1	1	1
36	1	1	1	1	1
37	1	1	1	1	1
38	1	1	1	1	1
39	1	1	1	1	1
40	1	1	1	1	1
41	1	1	1	1	1
42	1	1	1	1	1
43	1	1	1	1	1
44	1	1	1	1	1
45	1	1	1	1	1
46	1	1	1	1	1
47	1	1	1	1	1
48	1	1	1	1	1
49	1	1	1	1	1
50	1	1	1	1	1
51	1	1	1	1	1
52	1	1	1	1	1
53	1	1	1	1	1
54	1	1	1	1	1
55	1	1	1	1	1
56	1	1	1	1	1
57	1	1	1	1	1
58	1	1	1	1	1
59	1	1	1	1	1
60	1	1	1	1	1
61	1	1	1	1	1
62	1	1	1	1	1
63	1	1	1	1	1
64	1	1	1	1	1
65	1	1	1	1	1
66	1	1	1	1	1
67	1	1	1	1	1
68	1	1	1	1	1
69	1	1	1	1	1
70	1	1	1	1	1
71	1	1	1	1	1
72	1	1	1	1	1
73	1	1	1	1	1
74	1	1	1	1	1
75	1	1	1	1	1
76	1	1	1	1	1
77	1	1	1	1	1
78	1	1	1	1	1
79	1	1	1	1	1
80	1	1	1	1	1
81	1	1	1	1	1
82	1	1	1	1	1
83	1	1	1	1	1
84	1	1	1	1	1
85	1	1	1	1	1
86	1	1	1	1	1
87	1	1	1	1	1
88	1	1	1	1	1
89	1	1	1	1	1
90	1	1	1	1	1
91	1	1	1	1	1
92	1	1	1	1	1
93	1	1	1	1	1
94	1	1	1	1	1
95	1	1	1	1	1
96	1	1	1	1	1
97	1	1	1	1	1
98	1	1	1	1	1
99	1	1	1	1	1
100	1	1	1	1	1

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00

```
1 #include <cs50.h>
2
3 int countZeroes(int arr[], int low, int high) {
4     if (low > high) {
5         return 0;
6     }
7     int mid = low + (high - low) / 2;
8     if (arr[mid] == 0) {
9         return (high - mid + 1) + countZeroes(arr, low, mid - 1);
10    } else {
11        return countZeroes(arr, mid + 1, high);
12    }
13}
14
15 int main() {
16     int n;
17     scanf("%d", &n);
18     int arr[n];
19     for (int i = 0; i < n; i++) {
20         scanf("%d", &arr[i]);
21     }
22     int numZeroes = countZeroes(arr, 0, n - 1);
23     printf("%d\n", numZeroes);
24     return 0;
25 }
```

Input	Expected	Got
5 2 1 1 1 0 0	2	2 ✓
10 2 1 1 1 1 1 1 1 1	8	8 ✓
8 8 8 8	8	8 ✓

1-Number of Zeros in a Given Array

Started on: Tuesday, 9 September 2025, 12:11 PM

State: Finished

Completed on: Tuesday, 9 September 2025, 12:14 PM

Time taken: 2 min 44 secs

Marks: 1.00/1.00

Grade: 10.00 out of 10.00 (100%)

Question 1: Correct Marks 1.00 out of 1.00 [Flag question](#)

Problem Statement:

Given an array of 1s and 0s this has all 1s first followed by all 0s. Aim is to find the number of 0s. Write a program using Divide and Conquer to Count the number of zeroes in the given array.

Input Format:

First Line Contains Integer m – Size of array

Next m lines Contains m numbers – Elements of an array

Output Format:

First Line Contains Integer – Number of zeroes present in the given array.

Answer: (penalty regime: 0 %)

```
1. #include <iostream.h>
2.
3. int countZeroes(int arr[], int low, int high) {
4.     if (low > high) {
5.         return 0;
6.     }
7.     int mid = low + (high - low) / 2;
8.     if (arr[mid] == 0) {
9.         return (high - mid + 1) + countZeroes(arr, low, mid - 1);
10.    } else {
11.        return countZeroes(arr, mid + 1, high);
12.    }
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.
25. }
```

```
1. #include <iostream.h>
2.
3. int countZeroes(int arr[], int low, int high) {
4.     if (low > high) {
5.         return 0;
6.     }
7.     int mid = low + (high - low) / 2;
8.     if (arr[mid] == 0) {
9.         return (high - mid + 1) + countZeroes(arr, low, mid - 1);
10.    } else {
11.        return countZeroes(arr, mid + 1, high);
12.    }
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.
25. }
```

Closed DAA/Screenshot 2025-11-03
at 21.09.32.pdf at 1cb41deb11db6e

Undo