×

- General
- BASIC C PROGRAMMING
  - BASIC C PROGRAMMING-PR...
- Finding Time Complexity ...
  - Problem 1: Finding Complex...
  - Problem 2: Finding Complex...
  - Problem 3: Finding Complex...
  - Problem 4: Finding Complex...
  - Problem 5: Finding Complex...
- Divide and Conquer
  - 1-Number of Zeros in a Give...
  - 2-Majority Element
  - 3-Finding Floor Value
  - 4-Two Elements sum to x
  - 5-Implementation of Quick ...
- Greedy Algorithms
  - 1-G-Coin Problem
  - 2-G-Cookies Problem
  - 3-G-Burger Problem
  - 4-G-Array Sum max problem
  - 5-G-Product of Array eleme...
- Dynamic Programming
  - 1-DP-Playing with Numbers

1-Number of Zeros in a Given Array

Started on	Friday, 19 September 2025, 2:24 PM
State	Finished
Completed on	Tuesday, 18 November 2025, 4:59 PM
Time taken	60 days 2 hours
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1 Correct Mark 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 %)

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++){
     scanf("%d",&arr[i]);
    }
    int mid=n/2;
    int count=0;
    for(int i=0;i<mid;i++){
        if(arr[i]==0){
            count++;
        }
    }
    for(int j=mid;j<n;j++){
        if(arr[j]==0){
            count++;
        }
    }
    printf("%d",count);
}
```

	Input	Expected	Got	
✓	5 1 1 1 0 0	2	2	✓
✓	10 1 1 1 1 1 1 1 1 1 1	0	0	✓
✓	8 0 0 0 0 0 0 0 0	8	8	✓
✓	17 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0	2	2	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

×

- General
- BASIC C PROGRAMMING
- BASIC C PROGRAMMING-PR...
- Finding Time Complexity ...
- Problem 1: Finding Complex...
- Problem 2: Finding Complex...
- Problem 3: Finding Complex...
- Problem 4: Finding Complex...
- Problem 5: Finding Complex...
- Divide and Conquer
- 1-Number of Zeros in a Give...
- 2-Majority Element
- 3-Finding Floor Value
- 4-Two Elements sum to x
- 5-Implementation of Quick ...
- Greedy Algorithms
- 1-G-Coin Problem
- 2-G-Cookies Problem
- 3-G-Burger Problem
- 4-G-Array Sum max problem
- 5-G-Product of Array eleme...
- Dynamic Programming
- 1-DP-Playing with Numbers

2-Majority Element

Started on	Friday, 19 September 2025, 2:35 PM
State	Finished
Completed on	Friday, 3 October 2025, 2:09 PM
Time taken	13 days 23 hours
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1 Correct Mark 1.00 out of 1.00 Flag question

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

The majority element is the element that appears more than `⌊n / 2⌋` 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 == nums.length`
- $1 <= n <= 5 * 10^4$
- $-2^{31} <= nums[i] <= 2^{31} - 1$

For example:

Input	Result
3 3 2 3	3
7 2 2 1 1 1 2 2	2

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    int count=0;
    int value=0;
    for(int i=0;i<n;i++){
        if(count==0){
            value=arr[i];
        }
        else if(value==arr[i+1]){
            count++;
        }
        else{
            count--;
        }
    }
    printf("%d",value);
    }

```

	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

×

- General
- BASIC C PROGRAMMING
- BASIC C PROGRAMMING-PR...
- Finding Time Complexity ...
- Problem 1: Finding Complex...
- Problem 2: Finding Complex...
- Problem 3: Finding Complex...
- Problem 4: Finding Complex...
- Problem 5: Finding Complex...
- Divide and Conquer
- 1-Number of Zeros in a Give...
- 2-Majority Element
- 3-Finding Floor Value
- 4-Two Elements sum to x
- 5-Implementation of Quick ...
- Greedy Algorithms
- 1-G-Coin Problem
- 2-G-Cookies Problem
- 3-G-Burger Problem
- 4-G-Array Sum max problem
- 5-G-Product of Array eleme...
- Dynamic Programming
- 1-DP-Playing with Numbers

RAJALAKSHMI ENGINEERING COLLEGE

7 JAI VARDHAN U L 2024-CSE J2

Dashboard My courses

CS23331-DAA-2024-CSE / 3-Finding Floor Value

3-Finding Floor Value

Started on	Friday, 3 October 2025, 2:09 PM
State	Finished
Completed on	Friday, 10 October 2025, 2:14 PM
Time taken	7 days
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1 Correct Mark 1.00 out of 1.00 Flag question

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: (penalty regime: 0 %)

```
#include<stdio.h>
int findfloor(int arr[],int n,int x){
    int low=0;
    int high=n-1;
    int floorVal=0;
    while(low<=high){
        int mid=(low+high)/2;

        if(arr[mid]==x){
            return arr[mid];
        }else if(arr[mid]<x){
            floorVal=arr[mid];
            low=mid+1;
        }else{
            high=mid-1;
        }
    }
    return floorVal;
}
int main(){
    int n,x;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    scanf("%d",&x);
    int floorVal=findfloor(arr,n,x);
    printf("%d\n",floorVal);
    return 0;
}

```

	Input	Expected	Got	
✓	6 1 2 8 10 12 19 5	2	2	✓
✓	5 10 22 85 108 129 100	85	85	✓
✓	7 3 5 7 9 11 13 15 10	9	9	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

×

BASIC C PROGRAMMING

- BASIC C PROGRAMMING-PR...

Finding Time Complexity ...

- Problem 1: Finding Complex...
- Problem 2: Finding Complex...
- Problem 3: Finding Complex...
- Problem 4: Finding Complex...
- Problem 5: Finding Complex...

Divide and Conquer

- 1-Number of Zeros in a Give...
- 2-Majority Element
- 3-Finding Floor Value
- 4-Two Elements sum to x
- 5-Implementation of Quick ...

Greedy Algorithms

- 1-G-Coin Problem
- 2-G-Cookies Problem
- 3-G-Burger Problem
- 4-G-Array Sum max problem
- 5-G-Product of Array eleme...

Dynamic Programming

- 1-DP-Playing with Numbers
- 2-DP-Playing with chessboard

4-Two Elements sum to x

Started on	Friday, 3 October 2025, 2:16 PM
State	Finished
Completed on	Tuesday, 18 November 2025, 5:01 PM
Time taken	46 days 2 hours
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1 Correct Mark 1.00 out of 1.00 Flag question

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 – Element1

Second Line Contains Integer – Element2 (Element 1 and Elements 2 together sums to value "x")

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int findpair(int arr[],int low,int high,int x,int*a,int*b){
    if(low>=high){
        return 0;
    }
    int sum=arr[low]+arr[high];
    if(sum==x){
        *a=arr[low];
        *b=arr[high];
        return 1;
    }else if(sum<x){
        return findpair(arr,low+1,high,x,a,b);
    }else{
        return findpair(arr,low,high-1,x,a,b);
    }
}
int main(){
    int n,x;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    scanf("%d",&x);
    int a,b;
    if(findpair(arr,0,n-1,x,&a,&b)){
        printf("%d\n%d",a,b);
    }else{
        printf("No");
    }
    return 0;
}
```

	Input	Expected	Got	
✓	4 2 4 8 10 14	4 10	4 10	✓
✓	5 2 4 6 8 10 100	No	No	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

×
- BASIC C PROGRAMMING-PR...
- Finding Time Complexity ...
- Problem 1: Finding Complex...
- Problem 2: Finding Complex...
- Problem 3: Finding Complex...
- Problem 4: Finding Complex...
- Problem 5: Finding Complex...
- Divide and Conquer
- 1-Number of Zeros in a Give...
- 2-Majority Element
- 3-Finding Floor Value
- 4-Two Elements sum to x
- 5-Implementation of Quick ...
- Greedy Algorithms
- 1-G-Coin Problem
- 2-G-Cookies Problem
- 3-G-Burger Problem
- 4-G-Array Sum max problem
- 5-G-Product of Array eleme...
- Dynamic Programming
- 1-DP-Playing with Numbers
- 2-DP-Playing with chessboard
- 3-DP-Longest Common Sub...

Quiz navigation

1 ✓

Finish review

5-Implementation of Quick Sort

Started on	Friday, 10 October 2025, 2:15 PM
State	Finished
Completed on	Friday, 10 October 2025, 2:20 PM
Time taken	5 mins 46 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1 Correct Mark 1.00 out of 1.00 Flag question

Write a Program to Implement the Quick Sort Algorithm

Input Format:
The first line contains the no of elements in the list-n
The next n lines contain the elements.

Output:
Sorted list of elements

For example:

Input	Result
5 67 34 12 98 78	12 34 67 78 98

Answer:

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

int cmpfunc(const void*a,const void*b){
    return (*(int*)a-*(int*)b);
}
int main(){
    int n;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    qsort(arr,n,sizeof(int),cmpfunc);

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

	Input	Expected	Got	
✓	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98	✓
✓	10 1 56 78 90 32 56 11 10 90 114	1 10 11 32 56 56 78 90 90 114	1 10 11 32 56 56 78 90 90 114	✓
✓	12 9 8 7 6 5 4 3 2 1 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course