RAJALAKSHMI ENGINEERING COLLEGE

Approved by AICTE | Affiliated to Anna University | Accredited by NAAC

Department of Computer Science and Engineering

CS23331 Design and Analysis of Algorithms

III semester II Year (2023R)

Name of the Student : Soameshwaran D

Register Number : 2116240701520

Problem 1: Finding Complexity using Counter Method

Started on	Wednesday, 13 August 2025, 11:12 AM
State	Finished
Completed on	Wednesday, 13 August 2025, 11:16 AM
Time taken	3 mins 3 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

Convert the following algorithm into a program and find its time complexity using the counter method.

```
void function (int n)
{
    int i= 1;

    int s =1;

    while(s <= n)
    {
        i++;
        s += i;
     }
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Output:

Print the value of the counter variable

For example:

Input	Result
9	12

Open block draw

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int c=0;
void function (int n)
{
    int i= 1;
    c++;
    int s =1;
    c++;
    while(s <= n)
    {
        c++;
        i++;
        c++;

        s += i;
        c++;
    }
}
int main(){
    int n;
    scanf("%d",&n);
    c=0;
    function(n);
    printf("%d\n",c+1);
    return 0;
}
```


```c
    return 0;
}
```


	Input	Expected	Got	
✔	9	12	12	✔
✔	4	9	9	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

Dashboard My courses

CS23331-DAA-2024-CSE / Problem 2: Finding Complexity using Counter method

Problem 2: Finding Complexity using Counter method

Started on	Wednesday, 13 August 2025, 10:42 AM
State	Finished
Completed on	Wednesday, 13 August 2025, 10:51 AM
Time taken	9 mins 8 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

Convert the following algorithm into a program and find its time complexity using the counter method.

```
void func(int n)
{
   if(n==1)
   {
    printf("*");
   }
   else
   {
    for(int i=1; i<=n; i++)
    {
      for(int j=1; j<=n; j++)
      {
        printf("*");
        printf("*");
        break;
      }
    }
   }
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int main() {
    int n;
    int counter = 0;
    scanf("%d", &n);
    counter++;
    if (n == 1) {
        counter++;
    }
    else {
        counter += (n + 1);
        counter += 2 * n;
        counter += 2 * n;
    }
    printf("%d\n", counter);
    return 0;
}
```

	Input	Expected	Got	
✓	2	12	12	✓
✓	1000	5002	5002	✓
✓	143	717	717	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Dashboard My courses

CS23331-DAA-2024-CSE / Problem 3: Finding Complexity using Counter Method

Problem 3: Finding Complexity using Counter Method

Started on	Wednesday, 13 August 2025, 11:04 AM
State	Finished
Completed on	Wednesday, 13 August 2025, 11:09 AM
Time taken	4 mins 12 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

Convert the following algorithm into a program and find its time complexity using counter method.

```
Factor(num) {
{
   for (i = 1; i <= num;++i)
   {
    if (num % i== 0)
      {
       printf("%d ", i);
      }
    }
  }
```

```
  }
}
```

Note: No need of counter increment for declarations and scanf() and counter variable printf() statement.

Open block drawer

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer:

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int counter = 1;
    for (int i = 1; i <= n;++i){
        counter++;
        counter++;
        if(n % i== 0){
            counter++;
        }
    }
    printf("%d\n", counter);
    return 0;
}
```

	Input	Expected	Got	
✔	12	31	31	✔
✔	25	54	54	✔
✔	4	12	12	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Dashboard My courses

CS23331-DAA-2024-CSE / Problem 4: Finding Complexity using Counter Method

Problem 4: Finding Complexity using Counter Method

Started on	Wednesday, 13 August 2025, 10:52 AM
State	Finished
Completed on	Wednesday, 13 August 2025, 11:12 AM
Time taken	20 mins 8 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

Convert the following algorithm into a program and find its time complexity using counter method.

```
void function(int n)
{
    int c= 0;
    for(int i=n/2; i<n; i++)
        for(int j=1; j<n; j = 2 * j)
            for(int k=1; k<n; k = k * 2)
                c++;
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Open block dra

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer:

```c
#include <stdio.h>
int cc;
void function(int n)
{
   cc++;
   cc++;
    for(int i=n/2; i<n; i++){
        cc++;
        cc++;
        for(int j=1; j<n; j = 2 * j){
            cc++;
            cc++;
            for(int k=1; k<n; k = k * 2){
                cc++;
                cc++;
            }
        }
    }
}
int main(){
    int a;
    scanf("%d",&a);

    function(a);
    printf("%d\n",cc);
}
```

	Input	Expected	Got	
✔	4	30	30	✔
✔	10	212	212	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Problem 5: Finding Complexity using counter method

Started on	Wednesday, 13 August 2025, 11:16 AM
State	Finished
Completed on	Wednesday, 13 August 2025, 11:18 AM
Time taken	2 mins 19 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

Convert the following algorithm into a program and find its time complexity using counter method.

```
void reverse(int n)
{
   int rev = 0, remainder;
   while (n != 0)
    {
        remainder = n % 10;
        rev = rev * 10 + remainder;
        n/= 10;

    }
print(rev);
}
```

Open block d

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer:

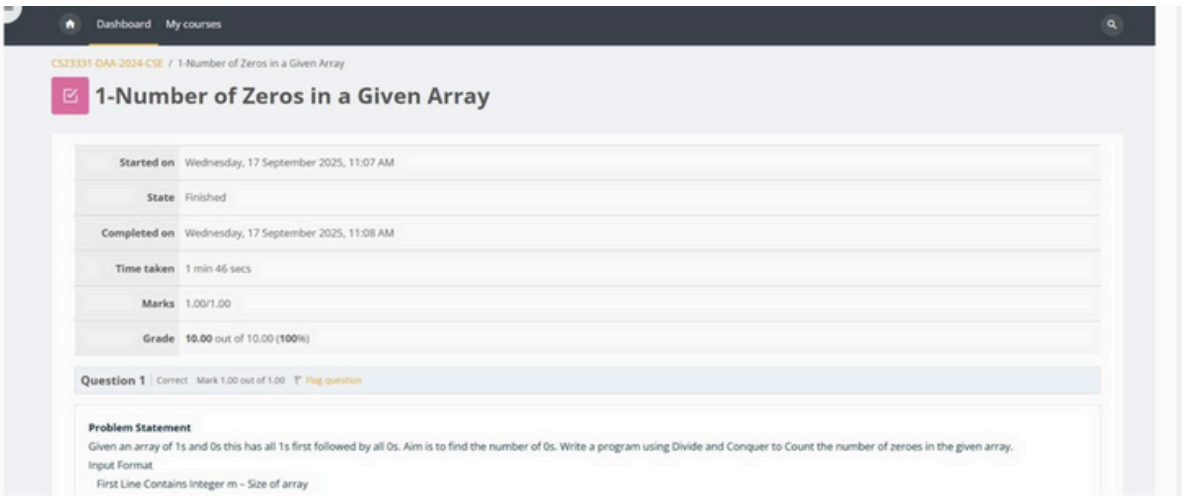
```c
#include <stdio.h>
int c=0;
void reverse(int n)
{
   int rev = 0,remainder;
   c++;
   while (n != 0)
    {
        c++;
        remainder = n % 10;
        c++;
        rev = rev * 10 + remainder;
        c++;
        n/= 10;
        c++;

    }
}
int main(){
    int a;
    scanf("%d\n",&a);
    c=0;
    reverse(a);
    printf("%d\n",c+2);
    return 0;
 }
```

	Input	Expected	Got	
✔	12	11	11	✔
✔	1234	19	19	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.



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

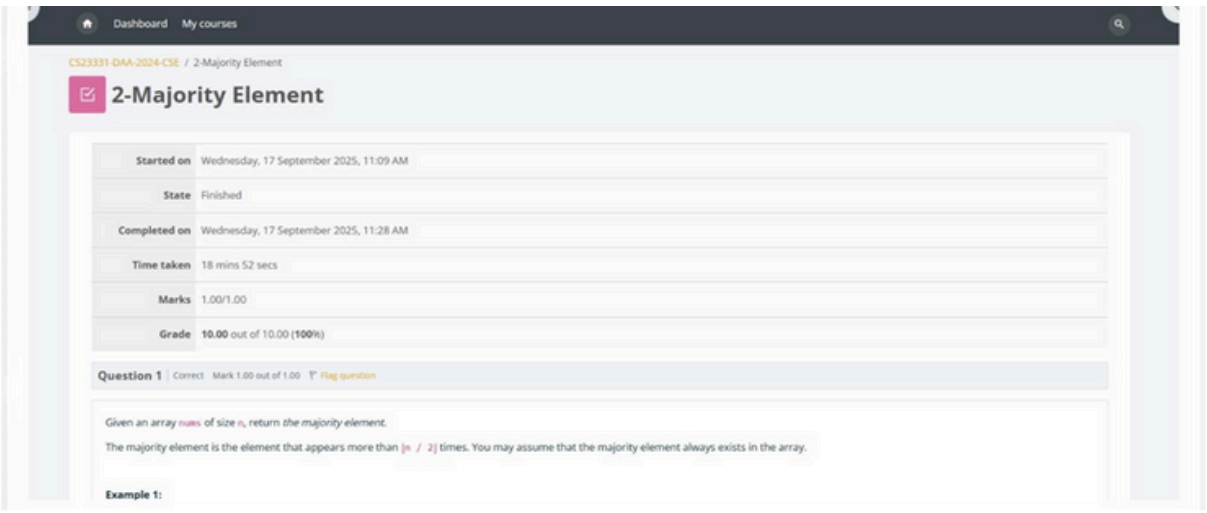
Output Format

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

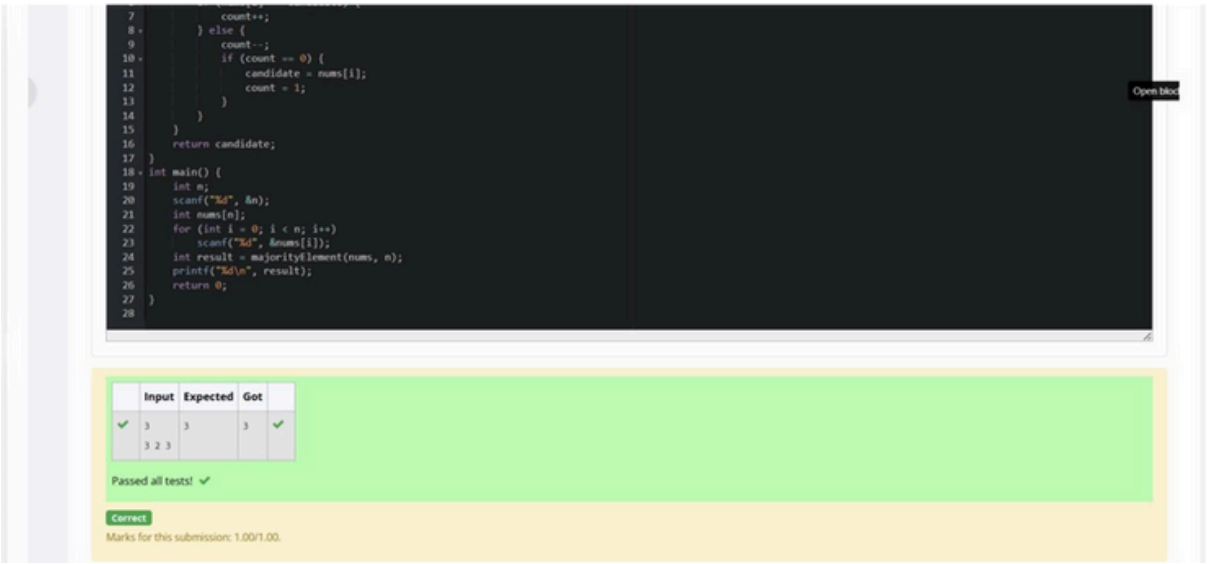
Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int firstZeroIndex(int arr[], int low, int high) {
    if (high >= low) {
        int mid = (low + high) / 2;
        if ((mid == 0 || arr[mid - 1] == 1) && arr[mid] == 0)
            return mid;
        if (arr[mid] == 1)
            return firstZeroIndex(arr, mid + 1, high);
        return firstZeroIndex(arr, low, mid - 1);
    }
    return -1;
}
int main() {
    int m;
    scanf("%d", &m);
    int arr[m];
    for (int i = 0; i < m; i++)
        scanf("%d", &arr[i]);
    int index = firstZeroIndex(arr, 0, m - 1);
    if (index == -1)
        printf("0\n");
    else
        printf("%d\n", m - index);
    return 0;
}
```

	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	2	2	✔


Dashboard
My courses
CS23331-DAA-2024-CSE / 2-Majority Element
2-Majority Element
Started on Wednesday, 17 September 2025, 11:09 AM
State Finished
Completed on Wednesday, 17 September 2025, 11:28 AM
Time taken 18 mins 52 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
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:


Example 1:
Input: nums = [3,2,3]
Output: 3
Example 2:
Input: nums = [2,2,1,1,1,2,2]
Output: 2
Open block drawer
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 %)
#include <stdio.h>
int majorityElement(int* nums, int n) {
int candidate = nums[0];
int count = 1;
for (int i = 1; i < n; i++) {
if (nums[i] == candidate) {
count++;


count++;
} else {
count--;
if (count == 0) {
candidate = nums[i];
count = 1;
}
}
}
return candidate;
}
int main() {
int n;
scanf("%d", &n);
int nums[n];
for (int i = 0; i < n; i++)
scanf("%d", &nums[i]);
int result = majorityElement(nums, n);
printf("%d\n", result);
return 0;
}
Input
Expected
Got
3
3 2 3
3
3
Passed all tests!
Correct
Marks for this submission: 1.00/1.00.

Dashboard My courses

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

3-Finding Floor Value

Started on	Wednesday, 17 September 2025, 11:28 AM
State	Finished
Completed on	Wednesday, 17 September 2025, 11:29 AM
Time taken	1 min 7 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

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

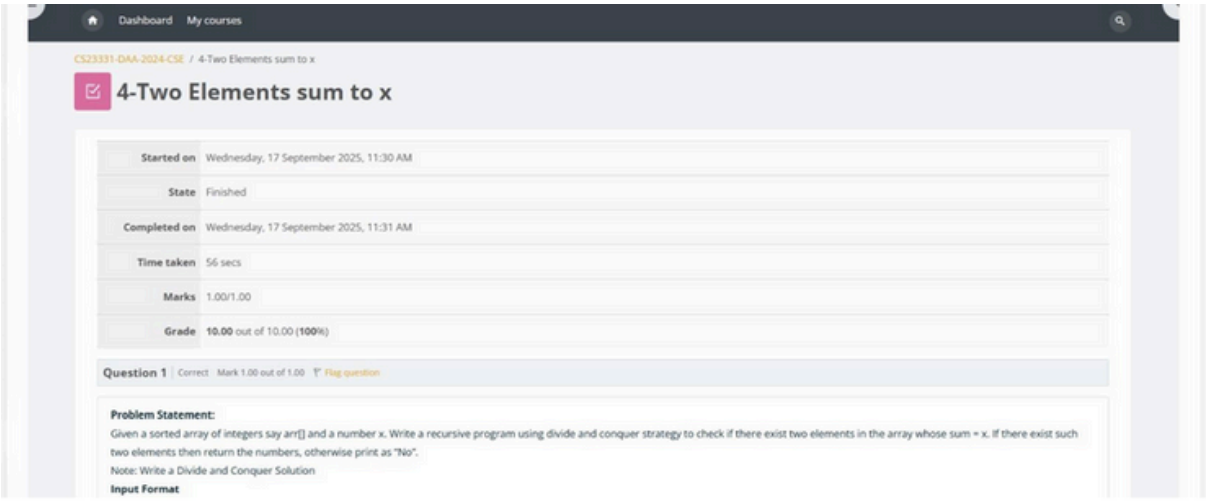
Output Format

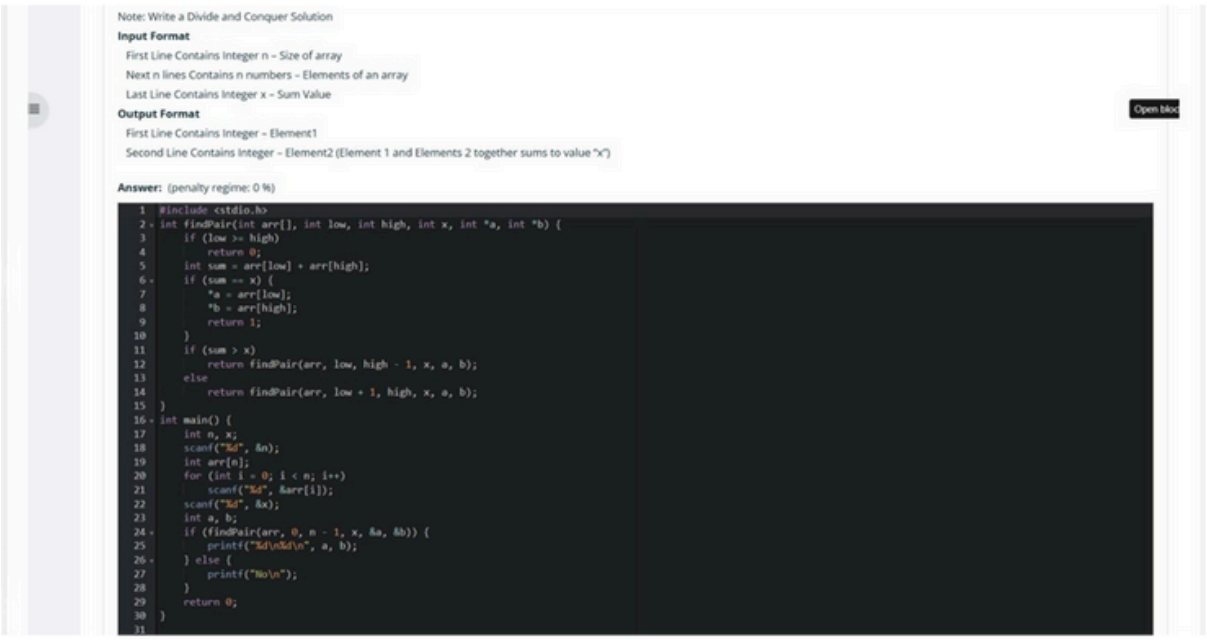
First Line Contains Integer – Floor value for x

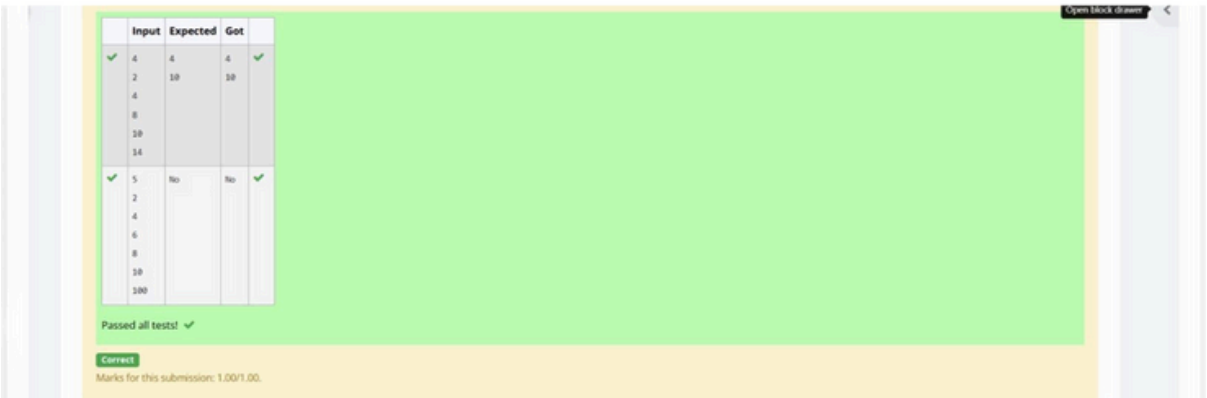
Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int findFloor(int arr[], int low, int high, int x) {
    if (low > high)
        return -1;

    int mid = (low + high) / 2;
    if (arr[mid] == x)
        return arr[mid];
    if (arr[mid] > x)
        return findFloor(arr, low, mid - 1, x);
    if (mid == high || arr[mid + 1] > x)
        return arr[mid];
    return findFloor(arr, mid + 1, high, x);
}
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, 0, n - 1, x);
    if (floorVal == -1)
        printf("No floor exists\n");
    else
        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	9	9	✔


Dashboard
My courses
CS23331-DAA-2024-CSE / 4-Two Elements sum to x
4-Two Elements sum to x
Started on Wednesday, 17 September 2025, 11:30 AM
State Finished
Completed on Wednesday, 17 September 2025, 11:31 AM
Time taken 56 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
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


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 %)
Open bloc
#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;
}
if (sum > x)
return findPair(arr, low, high - 1, x, a, b);
else
return findPair(arr, low + 1, high, 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\n", a, b);
} else {
printf("No\n");
}
return 0;
}


Open block drawer
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.

CS23331-DAA-2024-CSE / 5-Implementation of Quick Sort

5-Implementation of Quick Sort

Started on	Wednesday, 17 September 2025, 11:31 AM
State	Finished
Completed on	Wednesday, 17 September 2025, 11:32 AM
Time taken	1 min 18 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.

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

Output:
Sorted list of elements

Open block dra

For example:

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

Answer:

```c
#include <stdio.h>
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (arr[j] <= pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return i + 1;
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    quickSort(arr, 0, n - 1);
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
    return 0;
}
```

Open block drawer

	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.

Dashboard My courses

CS23331-DAA-2024-CSE / 1-G-Coin Problem

1-G-Coin Problem

Started on	Sunday, 17 August 2025, 7:35 PM
State	Finished
Completed on	Sunday, 17 August 2025, 7:42 PM
Time taken	6 mins 26 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 take value V and we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change.

Input Format:

Take an integer from stdin.

Take an integer from stdin.

Output Format:

print the integer which is change of the number.

Example Input :

64

Output:

4

Explanaton:

We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int v;
    scanf("%d",&v);
    int a[9]={1,2,5,10,20,50,100,500,1000};
    int s=0,c=0;
    for(int i=8;i>=0;i--){
        while(s+(a[i])<=v){
            s+=a[i];
            c++;
        }
    }
    printf("%d",c);
}
```

	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Dashboard My courses

CS23331-DAA-2024-CSE / 2-G-Cookies Problem

2-G-Cookies Problem

Started on	Tuesday, 19 August 2025, 9:25 PM
State	Finished
Completed on	Tuesday, 19 August 2025, 9:31 PM
Time taken	5 mins 51 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

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.

Each child i has a greed factor g[i], which is the minimum size of a cookie that the child will be content with; and each cookie j has a size s[j]. If s[j] >= g[i], we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and output the maximum number.

Example 1:

Example 1:

Input:

3

1 2 3

2

1 1

Output:

1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

You need to output 1.

Constraints:

1 <= g.length <= 3 * 10^4

0 <= s.length <= 3 * 10^4

1 <= g[i], s[j] <= 2^31 - 1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <stdlib.h>
int compare(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);
}
int findContentChildren(int* g, int gSize, int* s, int sSize) {
    qsort(g, gSize, sizeof(int), compare);
    qsort(s, sSize, sizeof(int), compare);
    int child = 0, cookie = 0;
    while (child < gSize && cookie < sSize) {
        if (s[cookie] >= g[child]) {
            child++;
        }
        cookie++;
    }
    return child;
}
int main() {
    int gSize, sSize;
    scanf("%d", &gSize);
    int g[gSize];
    for (int i = 0; i < gSize; i++) {
        scanf("%d", &g[i]);
    }
    scanf("%d", &sSize);
    int s[sSize];
    for (int i = 0; i < sSize; i++) {
        scanf("%d", &s[i]);
    }
    int result = findContentChildren(g, gSize, s, sSize);
    printf("%d\n", result);
    return 0;
}
```

	Input	Expected	Got	
✔	2 1 2 3 1 2 3	2	2	✔

Passed all tests! ✔

CS23331-DAA-2024-CSE / 3-G-Burger Problem

3-G-Burger Problem

Started on	Wednesday, 3 September 2025, 10:20 AM
State	Finished
Completed on	Wednesday, 3 September 2025, 10:32 AM
Time taken	12 mins 18 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

A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person needs to run a distance to burn out his calories.
If he has eaten c burgers with c calories each, then he has to run at least $3^1 * c$ kilometers to burn out the calories. For example, if he ate 3
burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 1 + 9 + 18 = 28$.
But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. Determine the minimum distance
he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm.Apply greedy approach to solve the problem.

Output Format

Print: Minimum number of kilometers needed to run to burn out the calories

Sample Input

```
3
5 10 7
```

Sample Output

```
76
```

For example:

Test	Input	Result
Test Case 1	3 1 3 2	18

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
int cmp(const void *a,const void *b)
{
    return (*(int*)b-*(int*)a);
}
int main(){
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    scanf("%d",&a[i]);
    qsort(a,n,sizeof(int),cmp);
    for(int i=0;i<n;i++)
    scanf("%d",&a[i]);
    qsort(a,n,sizeof(int),cmp);
    int md=0;
    for(int i=0;i<n;i++)
    md+= pow(n,i)*a[i];
    printf("%d",md);
}
```

Open block drawer

	Test	Input	Expected	Got	
✔	Test Case 1	3 1 3 2	18	18	✔
✔	Test Case 2	4 7 4 9 6	389	389	✔
✔	Test Case 3	3 5 10 7	76	76	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

Dashboard My courses

CS23331-DAA-2024-CSE / 4-G-Array Sum max problem

4-G-Array Sum max problem

Started on	Tuesday, 19 August 2025, 9:38 PM
State	Finished
Completed on	Tuesday, 19 August 2025, 9:40 PM
Time taken	1 min 31 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

Given an array of N integer, we have to maximize the sum of arr[i] * i, where i is the index of the element (i = 0, 1, 2, ..., N).Write an algorithm based on Greedy technique with a Complexity O(nlogn).

Input Format:

First line specifies the number of elements-n

The next n lines contain the array elements.

The next n lines contain the array elements.

Output Format:

Maximum Array Sum to be printed.

Sample Input:

5

2 5 3 4 0

Sample output:

40

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <stdlib.h>
int compare(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);
}
int main() {
    int n;
    scanf("%d", &n);
    int* arr = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    qsort(arr, n, sizeof(int), compare);

    long long result = 0;
    for (int i = 0; i < n; i++) {
        result += (long long)arr[i] * i;
    }
    printf("%lld\n", result);
    free(arr);
    return 0;
}
```

	Input	Expected	Got	
✔	5 2 5 3 4 0	40	40	✔
✔	10 2 2 2 4 4 3 3 5 5 5	191	191	✔
✔	2 45 3	45	45	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Dashboard My courses

CS23331-DAA-2024-CSE / 5-G-Product of Array elements-Minimum

5-G-Product of Array elements-Minimum

Started on	Tuesday, 19 August 2025, 9:40 PM
State	Finished
Completed on	Tuesday, 19 August 2025, 9:41 PM
Time taken	1 min 27 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

Given two arrays array_One[] and array_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs(1 element from each) is minimum. That is SUM (A[i] * B[i]) for all i is minimum.

For example:

Input	Result

Input	Result
3 1 2 3 4 5 6	28

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <stdlib.h>
int asc(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);
}
int desc(const void* a, const void* b) {
    return (*(int*)b - *(int*)a);
}
int main() {
    int n;
    scanf("%d", &n);
    int* array_One = malloc(n * sizeof(int));
    int* array_Two = malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &array_One[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &array_Two[i]);
    }
    qsort(array_One, n, sizeof(int), asc);
    qsort(array_Two, n, sizeof(int), desc);
    long long sum = 0;
    for (int i = 0; i < n; i++) {
        sum += (long long)array_One[i] * array_Two[i];
    }
    printf("%lld\n", sum);
    free(array_One);
    free(array_Two);
    return 0;
}
```

Open block dra

	Input	Expected	Got	
✔	3 1 2 3 4 5 6	28	28	✔
✔	4 7 5 1 2 1 3 4 1	22	22	✔
✔	5 20 10 30 10	590	590	✔

Input: *6*

Output:*6*

Explanation: *There are 6 ways to 6 represent number with 1 and 3*

1+1+1+1+1+1

3+3

1+1+1+3

1+1+3+1

1+3+1+1

3+1+1+1

Input Format

First Line contains the number n

Output Format

Print: The number of possible ways 'n' can be represented using 1 and 3

Sample Input

6

Sample Output

6

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
unsigned long long countWays(int n) {
    unsigned long long ways[n + 1];
    ways[0] = 1ULL;
    for (int i = 1; i <= n; i++) {
        ways[i] = ways[i - 1];
        if (i - 3 >= 0) {
            ways[i] += ways[i - 3];
        }
```

Dashboard My courses

CS23331-DAA-2024-CSE / 1-DP-Playing with Numbers

1-DP-Playing with Numbers

Started on	Wednesday, 8 October 2025, 10:06 AM
State	Finished
Completed on	Wednesday, 8 October 2025, 10:11 AM
Time taken	4 mins 48 secs
Grade	**10.00** out of 10.00 (**100**%)

Question 1 Correct Mark 10.00 out of 10.00 Flag question

Playing with Numbers:

Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram term, so he gave Sita a positive integer 'n' and two numbers 1 and 3. He asked her to find the possible ways by which the number n can be represented using 1 and 3.Write any efficient algorithm to find the possible ways.

Example 1:

Input: *6*

Output:*6*

Explanation: *There are 6 ways to 6 represent number with 1 and 3*

```c
        if (i - 3 >= 0) {
            ways[i] += ways[i - 3];
        }
    }
    return ways[n];
}
int main() {
    int n;
    scanf("%d", &n);
    unsigned long long result = countWays(n);
    printf("%llu\n", result);
    return 0;
}
```

Open block drawer

	Input	Expected	Got	
✓	6	6	6	✓
✓	25	8641	8641	✓
✓	100	24382819596721629	24382819596721629	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

Dashboard My courses

CS23331-DAA-2024-CSE / 2-DP-Playing with chessboard

2-DP-Playing with chessboard

Started on	Wednesday, 8 October 2025, 10:11 AM
State	Finished
Completed on	Wednesday, 8 October 2025, 10:19 AM
Time taken	7 mins 11 secs
Grade	**10.00** out of 10.00 (**100%**)

Question 1 | Correct | Mark 10.00 out of 10.00 | Flag question

Playing with Chessboard:

Ram is given with an n*n chessboard with each cell with a monetary value. Ram stands at the (0,0), that the position of the top left white rook. He is been given a task to reach the bottom right black rook position (n-1, n-1) constrained that he needs to reach the position by traveling the maximum monetary path under the condition that he can only travel one step right or one step down the board. Help ram to achieve it by providing an efficient DP algorithm.

Example:

Input

Input

3

1 2 4

2 3 4

8 7 1

Output:

19

Explanation:

Totally there will be 6 paths among that the optimal is

Optimal path value:1+2+8+7+1=19

Input Format

First Line contains the integer n

The next n lines contain the n*n chessboard values

Output Format

Print Maximum monetary value of the path

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int max(int a, int b) {
    return (a > b) ? a : b;
}
int main() {
    int n;
    scanf("%d", &n);
    int board[n][n];
    int dp[n][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &board[i][j]);
        }
    }
    dp[0][0] = board[0][0];
    for (int j = 1; j < n; j++) {
        dp[0][j] = dp[0][j-1] + board[0][j];
    }
    for (int i = 1; i < n; i++) {
        dp[i][0] = dp[i-1][0] + board[i][0];
    }
    for (int i = 1; i < n; i++) {
        for (int j = 1; j < n; j++) {
            dp[i][j] = board[i][j] + max(dp[i-1][j], dp[i][j-1]);
        }
    }
    printf("%d\n", dp[n-1][n-1]);
    return 0;
}
```

	Input	Expected	Got	
✔	3 1 2 4 2 3 4 8 7 1	19	19	✔
✔	3 1 3 1 1 5 1 4 2 1	12	12	✔
✔	4 1 1 3 4 1 5 7 8 2 3 4 6	28	28	✔

Given two strings find the length of the common longest subsequence(need not be contiguous) between the two.

Example:

s1: ggtabe

s2: tgatasb

s1	a	g	**g**	**t**	**a**	**b**
s2	**g**	x	**t**	x	**a**	y	**b**

The length is 4

Solveing it using Dynamic Programming

For example:

Input	Result
aab azb	2

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <string.h>
int max(int a, int b) {
    return (a > b) ? a : b;
}
int main() {
    char s1[1001], s2[1001];
    scanf("%s %s", s1, s2);
```

```c
int main() {
    char s1[1001], s2[1001];
    scanf("%s %s", s1, s2);
    int len1 = strlen(s1);
    int len2 = strlen(s2);
    int dp[len1 + 1][len2 + 1];
    for (int i = 0; i <= len1; i++) {
        dp[i][0] = 0;
    }
    for (int j = 0; j <= len2; j++) {
        dp[0][j] = 0;
    }
    for (int i = 1; i <= len1; i++) {
        for (int j = 1; j <= len2; j++) {
            if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = 1 + dp[i - 1][j - 1];
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            }
        }
    }
    printf("%d\n", dp[len1][len2]);
    return 0;
}
```

Open course index

	Input	Expected	Got	
✔	aab azb	2	2	✔
✔	ABCD ABCD	4	4	✔

Passed all tests! ✔

Correct

Dashboard My courses

CS23331-DAA-2024-CSE / 4-DP-Longest non-decreasing Subsequence

4-DP-Longest non-decreasing Subsequence

Started on	Wednesday, 8 October 2025, 10:28 AM
State	Finished
Completed on	Wednesday, 8 October 2025, 10:29 AM
Time taken	1 min 29 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

Problem statement:

Find the length of the Longest Non-decreasing Subsequence in a given Sequence.

Eg:

Output:6

Answer: (penalty regime: 0 %)

Open course index

```c
#include <stdio.h>
int max(int a, int b) {
    return (a > b) ? a : b;
}
int main() {
    int n;
    scanf("%d", &n);
    int sequence[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &sequence[i]);
    }
    int dp[n];
    for (int i = 0; i < n; i++) {
        dp[i] = 1;
    }
    for (int i = 1; i < n; i++) {
        for (int j = 0; j < i; j++) {
            if (sequence[j] <= sequence[i]) {
                dp[i] = max(dp[i], dp[j] + 1);
            }
        }
    }
    int result = 1;
    for (int i = 0; i < n; i++) {
        if (dp[i] > result) {
            result = dp[i];
        }
    }
    printf("%d\n", result);
    return 0;
}
```

Open course index

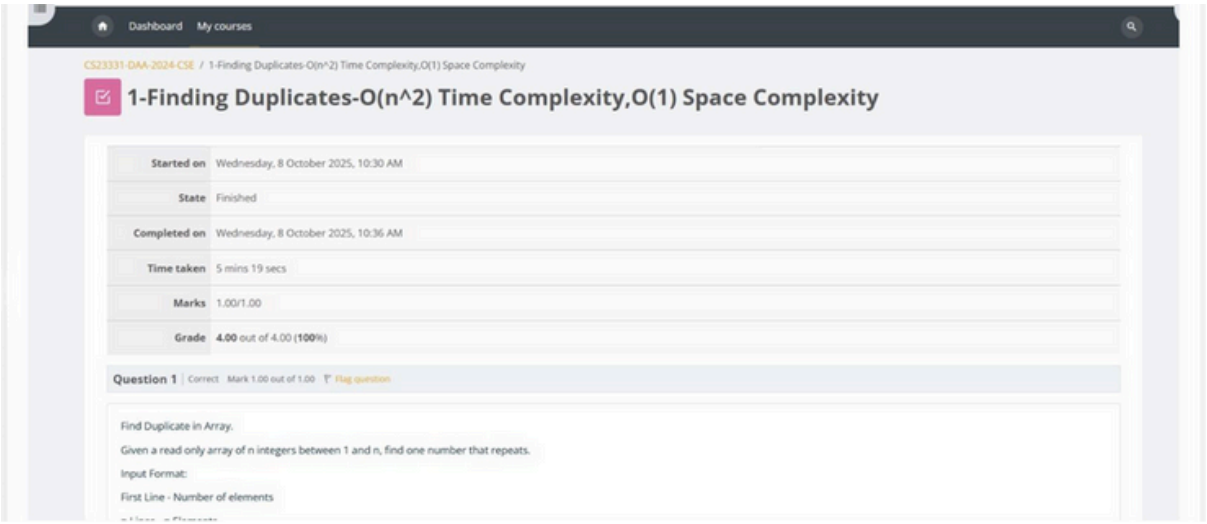
```c
        if (dp[i] > result) {
            result = dp[i];
        }
    }
    printf("%d\n", result);
    return 0;
}
```

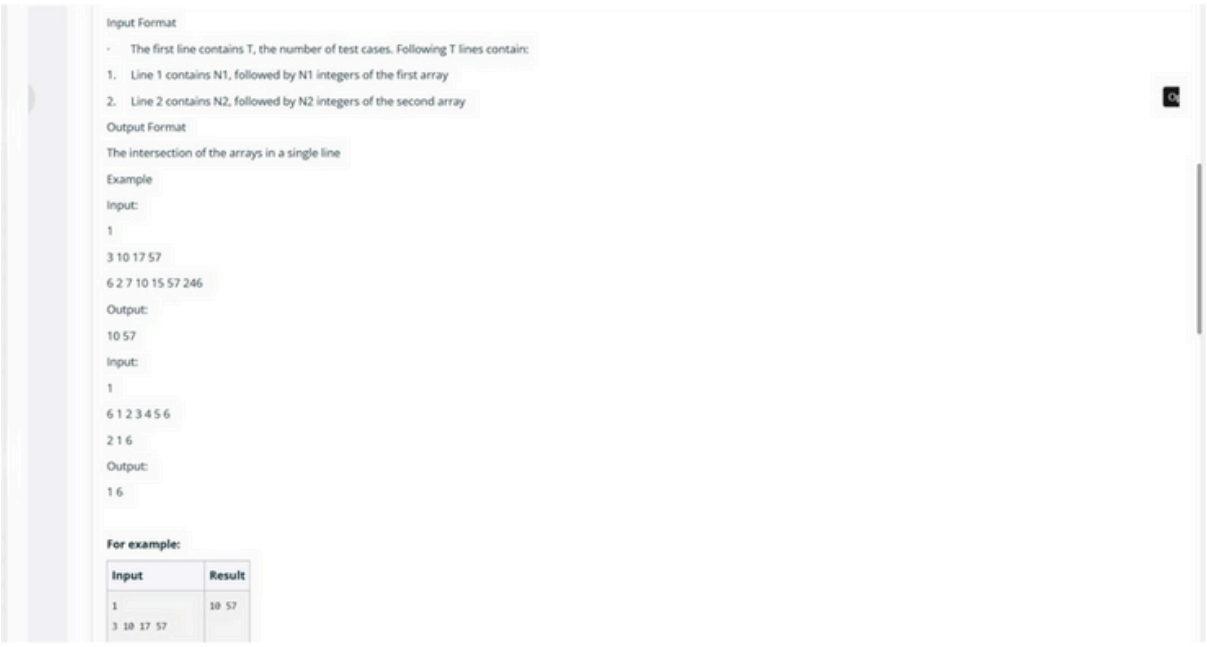
	Input	Expected	Got	
✔	9 -1 3 4 5 2 2 2 2 3	6	6	✔
✔	7 1 2 2 4 5 7 6	6	6	✔

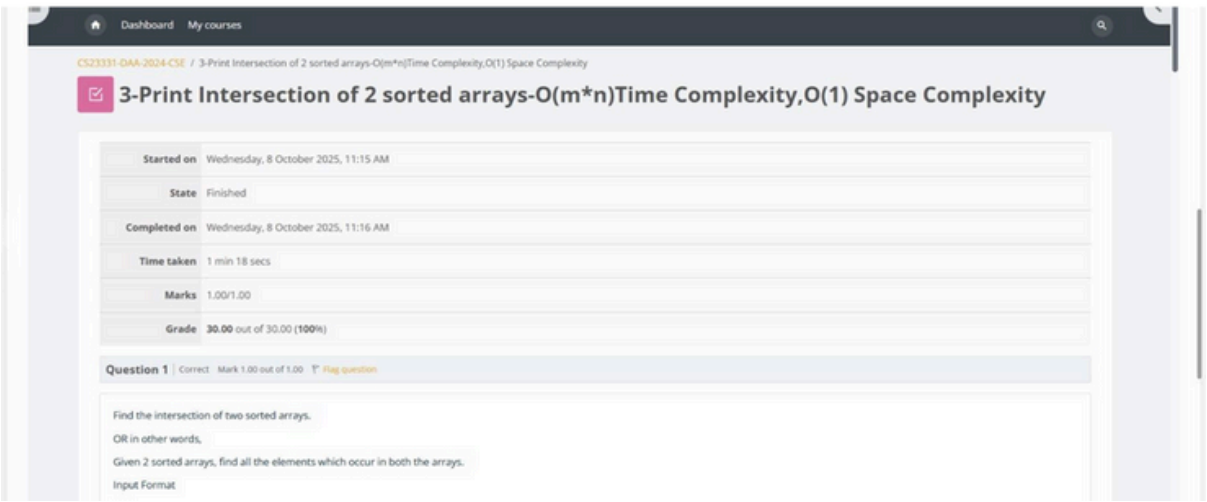
Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.


Dashboard My courses
CS23331-DAA-2024-CSE / 1-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity
1-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity
Started on Wednesday, 8 October 2025, 10:30 AM
State Finished
Completed on Wednesday, 8 October 2025, 10:36 AM
Time taken 5 mins 19 secs
Marks 1.00/1.00
Grade 4.00 out of 4.00 (100%)
Question 1 Correct Mark 1.00 out of 1.00 Flag question
Find Duplicate in Array.
Given a read only array of n integers between 1 and n, find one number that repeats.
Input Format:
First Line - Number of elements


Input Format
The first line contains T, the number of test cases. Following T lines contain:
1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array
Output Format
The intersection of the arrays in a single line
Example
Input:
1
3 10 17 57
6 2 7 10 15 57 246
Output:
10 57
Input:
1
6 1 2 3 4 5 6
2 1 6
Output:
1 6
For example:
Input
Result
1
3 10 17 57
10 57


Dashboard My courses
CS23331-DAA-2024-CSE / 3-Print Intersection of 2 sorted arrays-O(m*n)Time Complexity,O(1) Space Complexity
3-Print Intersection of 2 sorted arrays-O(m*n)Time Complexity,O(1) Space Complexity
Started on Wednesday, 8 October 2025, 11:15 AM
State Finished
Completed on Wednesday, 8 October 2025, 11:16 AM
Time taken 1 min 18 secs
Marks 1.00/1.00
Grade 30.00 out of 30.00 (100%)
Question 1 Correct Mark 1.00 out of 1.00 Flag question
Find the intersection of two sorted arrays.
OR in other words,
Given 2 sorted arrays, find all the elements which occur in both the arrays.
Input Format

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

Open bl

For example:

Input	Result
5 1 1 2 3 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int findDuplicate(int nums[], int n) {
    int slow = nums[0];
    int fast = nums[nums[0]];
    while (slow != fast) {
        slow = nums[slow];
        fast = nums[nums[fast]];
    }
    int slow2 = 0;
    while (slow2 != slow) {
        slow = nums[slow];
        slow2 = nums[slow2];
    }
    return slow;
}
int main() {
    int n;
    scanf("%d", &n);
    int nums[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }
    int duplicate = findDuplicate(nums, n);
    printf("%d\n", duplicate);
    return 0;
```

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

Open bl

For example:

Input	Result
5 1 1 2 3 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int findDuplicate(int nums[], int n) {
    int slow = nums[0];
    int fast = nums[nums[0]];
    while (slow != fast) {
        slow = nums[slow];
        fast = nums[nums[fast]];
    }
    int slow2 = 0;
    while (slow != slow2) {
        slow = nums[slow];
        slow2 = nums[slow2];
    }
    return slow;
}
int main() {
    int n;
    scanf("%d", &n);
    int nums[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }
    int duplicate = findDuplicate(nums, n);
    printf("%d\n", duplicate);
    return 0;
```

```c
    int duplicate = findDuplicate(nums, n);
    printf("%d\n", duplicate);
    return 0;
}
```

Open block d

	Input	Expected	Got	
✔	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✔
✔	5 1 2 3 4 4	4	4	✔
✔	5 1 1 2 3 4	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

```
    printf("%d\n", duplicate);
    return 0;
}
```

	Input	Expected	Got	
✔	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✔
✔	5 1 2 3 4 4	4	4	✔
✔	5 1 1 2 3 4	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

2-Finding Duplicates-O(n) Time Complexity,O(1) Space Complexity

Started on	Wednesday, 8 October 2025, 10:46 AM
State	Finished
Completed on	Wednesday, 8 October 2025, 10:47 AM
Time taken	1 min 8 secs
Marks	1.00/1.00
Grade	**4.00** out of 4.00 (**100**%)

Question 1 | Correct Mark 1.00 out of 1.00 Flag question

Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

Input Format:

First Line - Number of elements

Answer: (penalty regime: 0 %)

```
#include <stdio.h>
void findIntersection(int arr1[], int n1, int arr2[], int n2) {
    int i = 0, j = 0;
    while (i < n1 && j < n2) {
        if (arr1[i] == arr2[j]) {
            printf("%d ", arr1[i]);
            i++;
            j++;
        } else if (arr1[i] < arr2[j]) {
            i++;
        } else {
            j++;
        }
    }
    printf("\n");
}
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int n1;
        scanf("%d", &n1);
        int arr1[n1];
        for (int i = 0; i < n1; i++) {
            scanf("%d", &arr1[i]);
        }
        int n2;
        scanf("%d", &n2);
        int arr2[n2];
        for (int i = 0; i < n2; i++) {
            scanf("%d", &arr2[i]);
        }
        findIntersection(arr1, n1, arr2, n2);
    }
    return 0;
}
```

Open block drawer

Open block drawer

	Input	Expected	Got	
✔	1 3 10 17 57 6 2 7 10 15 57 246	10 57	10 57	✔
✔	1 6 1 2 3 4 5 6 2 1 6	1 6	1 6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Dashboard My courses

CS23331-DAA-2024-CSE / 4-Print Intersection of 2 sorted arrays-O(m+n)Time Complexity,O(1) Space Complexity

4-Print Intersection of 2 sorted arrays-O(m+n)Time Complexity,O(1) Space Complexity

Started on	Wednesday, 8 October 2025, 11:18 AM
State	Finished
Completed on	Wednesday, 8 October 2025, 11:19 AM
Time taken	1 min 5 secs
Marks	1.00/1.00
Grade	**30.00** out of 30.00 (**100**%)

Question 1 Correct Mark 1.00 out of 1.00 Flag question

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

Input Format

- The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Open block drawer

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

1 6

For example:

Input	Result
1 3 10 17 57 6	10 57

Answer: (penalty regime: 0 %)

Open block drawer

```c
#include <stdio.h>
void findIntersection(int arr1[], int n1, int arr2[], int n2) {
    int i = 0, j = 0;
    int first = 1;
    while (i < n1 && j < n2) {
        if (arr1[i] == arr2[j]) {
            if (!first) printf(" ");
            printf("%d", arr1[i]);
            first = 0;
            i++;
            j++;
        } else if (arr1[i] < arr2[j]) {
            i++;
        } else {
            j++;
        }
    }
    printf("\n");
}
int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int n1;
        scanf("%d", &n1);
        int arr1[n1];
        for (int i = 0; i < n1; i++) {
            scanf("%d", &arr1[i]);
        }
        int n2;
        scanf("%d", &n2);
        int arr2[n2];
        for (int i = 0; i < n2; i++) {
            scanf("%d", &arr2[i]);
        }
        findIntersection(arr1, n1, arr2, n2);
    }
    return 0;
}
```

Open block drawer

	Input	Expected	Got	
✔	1 3 10 17 57 6 2 7 10 15 57 246	10 57	10 57	✔
✔	1 6 1 2 3 4 5 6 2 1 6	1 6	1 6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Dashboard My courses

CS23331-DAA-2024-CSE / 5-Pair with Difference-O(n^2)Time Complexity,O(1) Space Complexity

5-Pair with Difference-O(n^2)Time Complexity,O(1) Space Complexity

Started on	Wednesday, 8 October 2025, 11:30 AM
State	Finished
Completed on	Wednesday, 8 October 2025, 11:31 AM
Time taken	54 secs
Marks	1.00/1.00
Grade	**4.00** out of 4.00 (**100**%)

Question 1 Correct Mark 1.00 out of 1.00 Flag question

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

Open block dra

For example:

Input	Result
3 1 3 5 4	1

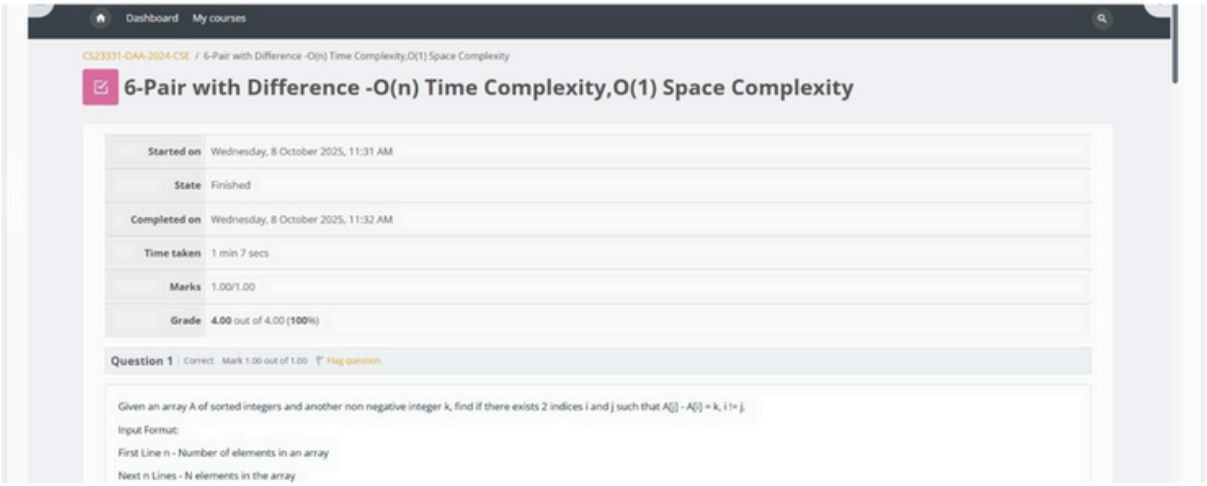
Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int findPairWithDiffK(int A[], int n, int k) {
    int i = 0, j = 1;
    while (j < n && i < n) {
        int diff = A[j] - A[i];
            if (diff == k && i != j) {
            return 1;
        } else if (diff < k) {
            j++;
        } else {
            i++;
            if (i == j) {
                j++;
            }
        }
    }
    return 0;
}
int main() {
    int n;
    scanf("%d", &n);
```

```c
}
int main() {
    int n;
    scanf("%d", &n);
    int A[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }
    int k;
    scanf("%d", &k);
    int result = findPairWithDiffK(A, n, k);
    printf("%d\n", result);
    return 0;
}
```

	Input	Expected	Got	
✔	3 1 3 5 4	1	1	✔
✔	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✔
✔	10 1 2 3 5 11 14 16 24 28 29 0	0	0	✔
✔	10 0 2 3 7 13 14 15 20 24 25 10	1	1	✔

Passed all tests! ✔



0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

Open block draw

For example:

Input	Result
3 1 3 5 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
int hasPairWithDiffK(int A[], int n, int k) {
    int i = 0, j = 1;
    while (j < n && i < n) {
        int diff = A[j] - A[i];
            if (diff == k && i != j) {
            return 1;
        } else if (diff < k) {
            j++;
        } else {
            i++;
            if (i == j) {
                j++;
            }
        }
    }
    return 0;
}
int main() {
    int n;
    scanf("%d", &n);
```

```c
}
int main() {
    int n;
    scanf("%d", &n);
    int A[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }
    int k;
    scanf("%d", &k);
    printf("%d\n", hasPairWithDiffK(A, n, k));
    return 0;
}
```

Open block d

	Input	Expected	Got	
✔	3 1 3 5 4	1	1	✔
✔	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✔
✔	10 1 2 3 5 11 14 16 24 28 29 0	0	0	✔
✔	10 0 2 3 7 13 14 15 20 24 25 10	1	1	✔

Passed all tests! ✔