Problem 1: Finding Complexity using Counter Method

Started on	Monday, 11 August 2025, 8:35 PM
State	Finished
Completed on	Monday, 11 August 2025, 8:39 PM
Time taken	4 mins 41 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.

Input:

A positive Integer n

Output:

Print the value of the counter variable

For example:

Input	Result
9	12

Answer: (penalty regime: 0 %)

```
#include<stdio.h>
int main(){
    int n;
    int count =0;
    scanf("%d",&n);
    int i =1;
    count++;
    int s= 1;
    count++;
    while(s <= n){
        count++;
        i++;
        count++;
        s+=i;
        count++;
    }
    count++;
    printf("%d\n",count);
    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

Problem 2: Finding Complexity using Counter method

Started on	Monday, 11 August 2025, 8:40 PM
State	Finished
Completed on	Monday, 11 August 2025, 8:44 PM
Time taken	3 mins 58 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;
    scanf("%d",&n);
    int counter =0;
    counter++;
    if(n!=1){
        for(int i=1;i<=n;i++){
            counter++;
            counter++;
            counter++;
            counter++;
            counter++;

        }
        counter++;
    }
    printf("%d",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.

Finish review

Back to Course

Problem 3: Finding Complexity using Counter Method

Started on	Monday, 11 August 2025, 8:44 PM
State	Finished
Completed on	Monday, 11 August 2025, 8:47 PM
Time taken	2 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

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.

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer:

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

Finish review

Back to Course

Problem 4: Finding Complexity using Counter Method

Started on	Monday, 11 August 2025, 8:47 PM
State	Finished
Completed on	Monday, 11 August 2025, 8:53 PM
Time taken	5 mins 40 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.

Input:

A positive Integer n

Output:

Print the value of the counter variable

Answer:

```c
#include<stdio.h>
#include<math.h>
int main(){
    int n;
    scanf("%d",&n);
    double a= 2.296;
    double b = -1.813;
    double c = 0.517;
    int counter = round(a* n* n + b * n + c);
    printf("%d",counter);
    return 0;
}
```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

Problem 5: Finding Complexity using counter method

Started on	Monday, 11 August 2025, 8:53 PM
State	Finished
Completed on	Monday, 11 August 2025, 8:57 PM
Time taken	4 mins 37 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);
}
```

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>
void reverse(int n){
    int rev = 0,remainder;
    int count = 1;
    while(n!=0){
        count++;
        remainder =  n%10;
        count++;
        rev = rev * 10 +remainder;
        count++;
        n/=10;
        count++;
    }
    count++;
    count++;
    printf("%d\n",count);

}
int main(){
    int n;
    scanf("%d", &n);
    reverse(n);
    return 0;
}
```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

1-Number of Zeros in a Given Array

Started on	Thursday, 18 September 2025, 10:26 AM
State	Finished
Completed on	Thursday, 18 September 2025, 10:35 AM
Time taken	8 mins 20 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 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,count = 0;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    for(int i =0;i<n;i++){
        if(arr[i]==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

2-Majority Element

Started on	Thursday, 18 September 2025, 10:38 AM
State	Finished
Completed on	Sunday, 28 September 2025, 4:15 PM
Time taken	10 days 5 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>
#include <stdlib.h>

int majorityElement(int nums[], int numsSize) {
    int count = 0;
    int candidate = 0;

    for (int i = 0; i < numsSize; i++) {
        if (count == 0) {
            candidate = nums[i];
        }
        count += (nums[i] == candidate) ? 1 : -1;
    }

    return candidate;
}

int main() {
    int n;
    scanf("%d", &n);

    int* nums = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }

    int result = majorityElement(nums, n);
    printf("%d\n", result);

    free(nums);
    return 0;
}

```

	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

3-Finding Floor Value

Started on	Sunday, 28 September 2025, 4:16 PM
State	Finished
Completed on	Sunday, 28 September 2025, 4:17 PM
Time taken	59 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;
    if (x >= arr[high])
        return arr[high];
    int mid = (low + high) / 2;
    if (arr[mid] == x)
        return arr[mid];
    if (mid > 0 && arr[mid - 1] <= x && x < arr[mid])
        return arr[mid - 1];
    if (x < arr[mid])
        return findFloor(arr, low, mid - 1, x);
    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 result = findFloor(arr, 0, n - 1, x);
    printf("%d\n", result);
    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

4-Two Elements sum to x

Started on	Thursday, 18 September 2025, 10:58 AM
State	Finished
Completed on	Sunday, 28 September 2025, 4:19 PM
Time taken	10 days 5 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 left, int right, int x, int* a, int* b) {
    if (left >= right)
        return 0;
    int sum = arr[left] + arr[right];
    if (sum == x) {
        *a = arr[left];
        *b = arr[right];
        return 1;
    } else if (sum < x) {
        return findPair(arr, left + 1, right, x, a, b);
    } else {
        return findPair(arr, left, right - 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\n", a, b);
    } else {
        printf("No\n");
    }
    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

5-Implementation of Quick Sort

Started on	Sunday, 28 September 2025, 4:20 PM
State	Finished
Completed on	Sunday, 28 September 2025, 4:22 PM
Time taken	2 mins 20 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>

void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}

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;
}
```

	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

1-G-Coin Problem

Started on	Friday, 22 August 2025, 6:05 PM
State	Finished
Completed on	Friday, 22 August 2025, 6:07 PM
Time taken	2 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

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.

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>
const int denom[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
const int n = sizeof(denom) / sizeof(denom[0]);

int main() {
    int V;
    if (scanf("%d", &V) != 1) {
        return 1;
    }

    int count = 0;

    for (int i = 0; i < n; i++) {
        if (V <= 0) break;

        count += V / denom[i];
        V %= denom[i];
    }

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

	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

2-G-Cookies Problem

Started on	Thursday, 21 August 2025, 10:44 AM
State	Finished
Completed on	Friday, 22 August 2025, 6:11 PM
Time taken	1 day 7 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

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:

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) {
    long long da = *(const long long *)a;
    long long db = *(const long long *)b;
    if (da < db) return -1;
    else if (da > db) return 1;
    return 0;
}
int maxContentChildren(long long g[], int n, long long s[], int m) {

    qsort(g, n, sizeof(long long), compare);
    qsort(s, m, sizeof(long long), compare);

    int i = 0, j = 0, count = 0;

    while (i < n && j < m) {
        if (g[i] <= s[j]) {
            count++;
            i++;
            j++;
        } else {
            j++;
        }
    }
    return count;
}

int main() {
    int n, m;

    if (scanf("%d", &n) != 1) return 1;
    long long g[n];
    for (int i = 0; i < n; i++) {
        scanf("%lld", &g[i]);
    }


    if (scanf("%d", &m) != 1) return 1;
    long long s[m];
    for (int j = 0; j < m; j++) {
        scanf("%lld", &s[j]);
    }


    int result = maxContentChildren(g, n, s, m);
    printf("%d\n", result);

    return 0;
}
```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

3-G-Burger Problem

Started on	Friday, 29 August 2025, 6:17 PM
State	Finished
Completed on	Friday, 29 August 2025, 6:29 PM
Time taken	12 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

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

Input Format

First Line contains the number of burgers

Second line contains calories of each burger which is n space-separate integers

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 %)

```
#include<stdio.h>
int main(){
    int b;
    scanf("%d",&b);
    int arr[b];
    for(int i=0;i<b;i++){
        scanf("%d",&arr[i]);
    }
    int i=0,d=0;
    while(i<(b-1)){
        if(arr[i]<arr[i+1]){
            d=arr[i+1];
            arr[i+1]=arr[i];
            arr[i]=d;
            i=0;
            continue;
        }
        i++;
    }
    int c=0;
    int s=0;
    for(int j=0;j<b;j++){
        s=1;
        for(int k=0;k<j;k++){
            s=s*b;
        }
        ;;
        c=c+(s*arr[j]);
    }
    printf("%d",c);
}
```

	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

4-G-Array Sum max problem

Started on	Friday, 22 August 2025, 6:14 PM
State	Finished
Completed on	Friday, 22 August 2025, 6:16 PM
Time taken	1 min 40 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.

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 cmp_asc(const void *a, const void *b) {
    long long x = *(const long long*)a;
    long long y = *(const long long*)b;
    if (x < y) return -1;
    else if (x > y) return 1;
    return 0;
}

int main() {
    int n;
    if (scanf("%d", &n) != 1) return 1;

    long long *arr = malloc(n * sizeof(long long));
    if (!arr) return 1;

    for (int i = 0; i < n; i++) {
        scanf("%lld", &arr[i]);
    }


    qsort(arr, n, sizeof(long long), cmp_asc);

    long long total = 0;

    for (int i = 0; i < n; i++) {
        total += arr[i] * i;
    }

    printf("%lld\n", total);

    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.

Finish review

Back to Course

5-G-Product of Array elements-Minimum

Started on	Friday, 22 August 2025, 6:16 PM
State	Finished
Completed on	Friday, 22 August 2025, 6:18 PM
Time taken	1 min 43 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
3 1 2 3 4 5 6	28

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <stdlib.h>
int cmp_asc(const void* a, const void* b) {
    int x = *(const int*)a;
    int y = *(const int*)b;
    return (x > y) - (x < y);
}
int cmp_desc(const void* a, const void* b) {
    int x = *(const int*)a;
    int y = *(const int*)b;
    return (y > x) - (y < x);
}

int main() {
    int n;
    if (scanf("%d", &n) != 1) return 1;

    int *A = malloc(n * sizeof(int));
    int *B = malloc(n * sizeof(int));
    if (!A || !B) return 1;

    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &B[i]);
    }
    qsort(A, n, sizeof(int), cmp_asc);
    qsort(B, n, sizeof(int), cmp_desc);

    long long result = 0;
    for (int i = 0; i < n; i++) {
        result += (long long)A[i] * B[i];
    }

    printf("%lld\n", result);

    free(A);
    free(B);
    return 0;
}
```

	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 40 8 9 4 3 10	590	590	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

1-DP-Playing with Numbers

Started on	Thursday, 9 October 2025, 10:33 AM
State	Finished
Completed on	Thursday, 9 October 2025, 10:45 AM
Time taken	12 mins 33 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*

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 dp[n + 1];
    dp[0] = 1;
    for (int i = 1; i <= n; i++) {
        dp[i] = 0;
        if (i >= 1)
            dp[i] += dp[i - 1];
        if (i >= 3)
            dp[i] += dp[i - 3];
    }
    return dp[n];
}

int main() {
    int n;
    scanf("%d", &n);
    printf("%llu\n", countWays(n));
    return 0;
}
```

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

Passed all tests! ✔

Correct

Marks for this submission: 10.00/10.00.

Finish review

Back to Course

2-DP-Playing with chessboard

Started on	Thursday, 9 October 2025, 10:46 AM
State	Finished
Completed on	Thursday, 9 October 2025, 10:58 AM
Time taken	12 mins 2 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

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 main() {
    int n;
    scanf("%d", &n);
    int board[n][n], 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 i = 1; i < n; i++)
        dp[i][0] = dp[i - 1][0] + board[i][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++)
        for (int j = 1; j < n; j++)
            dp[i][j] = board[i][j] + (dp[i - 1][j] > dp[i][j - 1] ? 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 1 6 9 0	28	28	✔

Passed all tests! ✔

Correct

Marks for this submission: 10.00/10.00.

Finish review

Back to Course

3-DP-Longest Common Subsequence

Started on	Thursday, 9 October 2025, 10:59 AM
State	Finished
Completed on	Thursday, 9 October 2025, 11:06 AM
Time taken	7 mins 15 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 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[1000], s2[1000];
    scanf("%s", s1);
    scanf("%s", s2);
    int m = strlen(s1), n = strlen(s2);
    int dp[m + 1][n + 1];
    for (int i = 0; i <= m; i++)
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (s1[i - 1] == s2[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;
            else
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
        }
    printf("%d\n", dp[m][n]);
    return 0;
}

```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

4-DP-Longest non-decreasing Subsequence

Started on	Thursday, 9 October 2025, 11:06 AM
State	Finished
Completed on	Thursday, 9 October 2025, 11:11 AM
Time taken	5 mins 9 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:

Input:9

Sequence:[-1,3,4,5,2,2,2,2,3]

the subsequence is [-1,2,2,2,2,3]

Output:6

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 arr[n], dp[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    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 (arr[i] >= arr[j])
                dp[i] = max(dp[i], dp[j] + 1);
    int result = 0;
    for (int i = 0; i < n; i++)
        result = max(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.

Finish review

Back to Course

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

Started on	Thursday, 9 October 2025, 11:13 AM
State	Finished
Completed on	Thursday, 9 October 2025, 11:21 AM
Time taken	8 mins 7 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

n Lines - n Elements

Output Format:

Element x - That is repeated

For example:

Input	Result
5 1 1 2 3 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int findDuplicate(int arr[], int n) {
    int slow = arr[0], fast = arr[0];
    do {
        slow = arr[slow];
        fast = arr[arr[fast]];
    } while (slow != fast);
    fast = arr[0];
    while (slow != fast) {
        slow = arr[slow];
        fast = arr[fast];
    }
    return slow;
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    printf("%d\n", findDuplicate(arr, n));
    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.

Finish review

Back to Course

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

Started on	Thursday, 9 October 2025, 11:21 AM
State	Finished
Completed on	Thursday, 9 October 2025, 11:32 AM
Time taken	10 mins 29 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

n Lines - n Elements

Output Format:

Element x - That is repeated

For example:

Input	Result
5 1 1 2 3 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int findDuplicate(int arr[], int n) {
    int slow = arr[0], fast = arr[0];
    do {
        slow = arr[slow];
        fast = arr[arr[fast]];
    } while (slow != fast);
    fast = arr[0];
    while (slow != fast) {
        slow = arr[slow];
        fast = arr[fast];
    }
    return slow;
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    printf("%d\n", findDuplicate(arr, n));
    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.

Finish review

Back to Course

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

Started on	Monday, 13 October 2025, 9:37 PM
State	Finished
Completed on	Monday, 13 October 2025, 9:52 PM
Time taken	15 mins 43 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

· 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 6 2 7 10 15 57 246	10 57

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main() {
    int T;
    scanf("%d", &T);

    while (T--) {
        int N1, N2;
        scanf("%d", &N1);
        int arr1[N1];
        for (int i = 0; i < N1; i++) {
            scanf("%d", &arr1[i]);
        }

        scanf("%d", &N2);
        int arr2[N2];
        for (int i = 0; i < N2; i++) {
            scanf("%d", &arr2[i]);
        }

        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");
    }

    return 0;
}
```

	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.

Finish review

Back to Course

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

Started on	Monday, 13 October 2025, 9:45 PM
State	Finished
Completed on	Monday, 13 October 2025, 9:53 PM
Time taken	8 mins 38 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

· 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 6 2 7 10 15 57 246	10 57

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

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]);
        }

        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");
    }

    return 0;
}
```

	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.

Finish review

Back to Course

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

Started on	Monday, 13 October 2025, 9:47 PM
State	Finished
Completed on	Monday, 13 October 2025, 10:07 PM
Time taken	19 mins 30 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

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

For example:

Input	Result
3 1 3 5 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main() {
    int n, k;
    scanf("%d", &n);
    int A[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }
    scanf("%d", &k);
    int i = 0, j = 1, found = 0;
    while (i < n && j < n) {
        if (i != j && A[j] - A[i] == k) {
            found = 1;
            break;
        } else if (A[j] - A[i] < k) {
            j++;
        } else {
            i++;
        }
    }
    printf("%d\n", found);
    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! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course

6-Pair with Difference -O(n) Time Complexity,O(1) Space Complexity

Started on	Monday, 13 October 2025, 9:49 PM
State	Finished
Completed on	Monday, 13 October 2025, 10:08 PM
Time taken	18 mins 23 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

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

For example:

Input	Result
3 1 3 5 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main() {
    int n, k;
    scanf("%d", &n);
    int A[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }
    scanf("%d", &k);
    int i = 0, j = 1;
    while (i < n && j < n) {
        if (i != j && A[j] - A[i] == k) {
            printf("1\n");
            return 0;
        } else if (A[j] - A[i] < k) {
            j++;
        } else {
            i++;
            if (i == j) j++;
        }
    }
    printf("0\n");
    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! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Back to Course