Started on	Tuesday, 19 November 2024, 9:49 PM
State	Finished
Completed on	Tuesday, 19 November 2024, 9:50 PM
Time taken	20 secs
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 10.00 out of 10.00

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>
int main() {
    long long n;
    scanf("%lld", &n);
    long long dp[n + 1];
    dp[0] = 1;
    if (n >= 1) dp[1] = 1;
    if (n >= 2) dp[2] = 1;
    if (n >= 3) dp[3] = 2;
    for (long long i = 4; i <= n; i++) {
        dp[i] = dp[i - 1];
        if (i - 3 >= 0) {
            dp[i] += dp[i - 3];
        }
    }
    printf("%lld\n", dp[n]);
    return 0;
}
```

	Input	Expected	Got	
✔	6	6	6	✔
✔	25	8641	8641	✔

	Input	Expected	Got	
✔	100	24382819596721629	24382819596721629	✔

Passed all tests! ✔

Correct

Marks for this submission: 10.00/10.00.

◄ 5-Implementation of Quick Sort

Jump to...

2-DP-Playing with chessboard ►

Started on	Friday, 16 August 2024, 1:41 PM
State	Finished
Completed on	Friday, 16 August 2024, 1:59 PM
Time taken	17 mins 34 secs
Marks	1.00/1.00
Grade	**4.00** out of 4.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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>
#include <stdlib.h>
int findDuplicate(int arr[], int n) {
    int *marker = (int *)calloc(n + 1, sizeof(int));
    for (int i = 0; i < n; i++) {
        if (marker[arr[i]] == 1) {
            free(marker);
            return arr[i];
        }
        marker[arr[i]] = 1;
    }

    free(marker);
    return -1;
}

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]);
    }
    int duplicate = findDuplicate(arr, n);
    if (duplicate != -1) {
        printf("%d\n", duplicate);
    } else {
        printf("No duplicate found\n");
    }


    free(arr);

    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	✔

	Input	Expected	Got	
✔	5 1 1 2 3 4	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 4-DP-Longest non-decreasing Subsequence

Jump to...

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

Started on	Friday, 23 August 2024, 1:36 PM
State	Finished
Completed on	Friday, 23 August 2024, 1:41 PM
Time taken	5 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

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>

int min_coins_for_change(int V) {
    int denominations[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
    int num_denominations = sizeof(denominations) / sizeof(denominations[0]);
    int count = 0;
    int remaining_amount = V;
    for (int i = 0; i < num_denominations; i++) {
        if (remaining_amount == 0) {
            break;
        }
        int num_notes = remaining_amount / denominations[i];
        count += num_notes;
        remaining_amount -= num_notes * denominations[i];
    }

    return count;
}

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

	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Problem 5: Finding Complexity using counter method

Jump to...

2-G-Cookies Problem ►

Started on	Friday, 30 August 2024, 1:37 PM
State	Finished
Completed on	Friday, 20 September 2024, 1:34 PM
Time taken	20 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

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 %)

```
#include <stdio.h>
int cz(int arr[], int s, int e) {
    if (s>e) {
        return 0;
    }
    if (s==e) {
        return arr[s] == 0 ? 1 : 0;
    }
    int mid=s+ (e- s)/2;
    if (arr[mid] == 0){
        return (e-mid+1)+cz(arr,s,mid-1);
    } else {
        return cz(arr,mid+1, e);
    }
}
int main() {
    int m;
    scanf("%d", &m);
```

	Input	Expected	Got	
✔	5 1 1 1 0 0	2	2	✔
✔	10 1 1 1 1 1 1 1 1 1 1	0	0	✔

	Input	Expected	Got	
✔	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.

◄ 5-G-Product of Array elements-Minimum

Jump to...

2-Majority Element ►

Started on	Tuesday, 19 November 2024, 9:50 PM
State	Finished
Completed on	Tuesday, 19 November 2024, 9:51 PM
Time taken	34 secs
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 10.00 out of 10.00

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 %)

```
#include <stdio.h>
#define MAX_N 100
int maxPathValue(int n, int board[MAX_N][MAX_N]) {
    int dp[MAX_N][MAX_N];
    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] + (dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j - 1]);
        }
    }
```

	Input	Expected	Got	
✔	3 1 2 4 2 3 4 8 7 1	19	19	✔

	Input	Expected	Got	
✔	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.

◄ 1-DP-Playing with Numbers

Jump to...

3-DP-Longest Common Subsequence ►

Started on	Friday, 16 August 2024, 2:00 PM
State	Finished
Completed on	Friday, 16 August 2024, 2:11 PM
Time taken	10 mins 50 secs
Marks	1.00/1.00
Grade	**4.00** out of 4.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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 %)

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

	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.

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

Jump to...

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

Started on	Friday, 23 August 2024, 1:42 PM
State	Finished
Completed on	Friday, 23 August 2024, 2:03 PM
Time taken	20 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

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) {
    return (*(int*)a - *(int*)b);
}
int find_content_children(int* g, int gSize, int* s, int sSize) {
    qsort(g, gSize, sizeof(int), compare);
    qsort(s, sSize, sizeof(int), compare);
    int child_i = 0;
    int cookie_i = 0;
    while (child_i < gSize && cookie_i < sSize) {
        if (s[cookie_i] >= g[child_i]) {
            child_i++;
        }
        cookie_i++;
    }
    return child_i;
}
int main() {
    int num_children, num_cookies;
    scanf("%d", &num_children);
    int* g = (int*)malloc(num_children * sizeof(int));
    for (int i = 0; i < num_children; i++) {
        scanf("%d", &g[i]);
    }
    scanf("%d", &num_cookies);
    int* s = (int*)malloc(num_cookies * sizeof(int));
    for (int i = 0; i < num_cookies; i++) {
        scanf("%d", &s[i]);
    }
    printf("%d\n", find_content_children(g, num_children, s, num_cookies));
    free(g);
    free(s);

    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.

◄ 1-G-Coin Problem

Jump to...

3-G-Burger Problem ►

Started on	Friday, 4 October 2024, 1:34 PM
State	Finished
Completed on	Friday, 4 October 2024, 1:35 PM
Time taken	1 min 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

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 candidate = nums[0];
    int count = 1;
    for (int i = 1; i < numsSize; i++) {
        if (count == 0) {
            candidate = nums[i];
            count = 1;
        } else if (nums[i] == candidate) {
            count++;
        } else {
            count--;
        }
    }
    return candidate;
}
int main() {
    int numsSize;
    scanf("%d", &numsSize);
    int* nums = (int*)malloc(numsSize * sizeof(int));
    if (nums == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }
    for (int i = 0; i < numsSize; i++) {
        scanf("%d", &nums[i]);
    }
    printf("%d\n", majorityElement(nums, numsSize));
    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.

◄ 1-Number of Zeros in a Given Array

Jump to...

3-Finding Floor Value ►

Started on	Tuesday, 19 November 2024, 9:51 PM
State	Finished
Completed on	Tuesday, 19 November 2024, 9:51 PM
Time taken	28 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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>
#define MAX_LEN 100

int lcsLength(const char *s1, const char *s2) {
    int len1 = strlen(s1);
    int len2 = strlen(s2);
    int dp[MAX_LEN + 1][MAX_LEN + 1];
    for (int i = 0; i <= len1; i++) {
        for (int j = 0; j <= len2; 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] = dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j - 1];
            }
        }
    }
    return dp[len1][len2];
}
int main() {
    char s1[MAX_LEN + 1], s2[MAX_LEN + 1];
    scanf("%s", s1);
    scanf("%s", s2);
    int result = lcsLength(s1, s2);
    printf("%d\n", result);
    return 0;
}

```

	Input	Expected	Got	
✔	aab azb	2	2	✔

	Input	Expected	Got	
✔	ABCD ABCD	4	4	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 2-DP-Playing with chessboard

Jump to...

4-DP-Longest non-decreasing Subsequence ►

Started on	Friday, 4 October 2024, 1:37 PM
State	Finished
Completed on	Friday, 4 October 2024, 1:37 PM
Time taken	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

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 n, int x) {
    int low = 0, high = n - 1;
    int result = -1;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == x) {
            return arr[mid];
        } else if (arr[mid] < x) {
            result = arr[mid];
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return result;
}
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 floor = findFloor(arr, n, x);
    printf("%d\n", floor);
    return 0;
}
```

	Input	Expected	Got	
✔	6 1 2 8 10 12 19 5	2	2	✔
✔	5 10 22 85 108 129 100	85	85	✔

	Input	Expected	Got	
✔	7 3 5 7 9 11 13 15 10	9	9	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 2-Majority Element

Jump to...

4-Two Elements sum to x ►

Started on	Friday, 23 August 2024, 2:05 PM
State	Finished
Completed on	Friday, 23 August 2024, 2:45 PM
Time taken	40 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

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 i burgers with c calories each, then he has to run at least $3^i * c$ 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.

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 %)

```c
#include<stdio.h>
#include<math.h>
int main()
{
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    scanf("%d",&a[i]);
    for (int i = 0; i < n-1; i++) {

        for ( int j = i + 1; j < n; j++) {
            int t;

            if (a[i] < a[j]) {
                t = a[i];
                a[i] = a[j];
                a[j] = t;
            }
        }
    }
    int sum=0,h;
    for(int i=0;i<n;i++)
    {
        h=pow(n,i);
        sum+=h*a[i];
    }
    printf("%d",sum);
}
```

	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.

◄ 2-G-Cookies Problem

Jump to...

4-G-Array Sum max problem ►

Started on	Friday, 25 October 2024, 1:35 PM
State	Finished
Completed on	Friday, 25 October 2024, 1:43 PM
Time taken	8 mins 2 secs
Marks	1.00/1.00
Grade	**30.00** out of 30.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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>

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]) {
            i++;
        } else if (arr1[i] > arr2[j]) {
            j++;
        } else {

            printf("%d ", arr1[i]);
            i++;
            j++;
        }
    }
}

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);
        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.

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

Jump to...

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

Started on	Tuesday, 19 November 2024, 9:52 PM
State	Finished
Completed on	Tuesday, 19 November 2024, 9:52 PM
Time taken	30 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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 longest_non_decreasing_subsequence(int sequence[], int n) {
    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] = (dp[i] > dp[j] + 1) ? dp[i] : dp[j] + 1;
            }
        }
    }
    int max_len = dp[0];
    for (int i = 1; i < n; i++) {
        if (dp[i] > max_len) {
            max_len = dp[i];
        }
    }
    return max_len;
}
int main() {
    int sequence[] = {-1, 3, 4, 5, 2, 2, 2, 2, 3};
    int n = sizeof(sequence) / sizeof(sequence[0]);
    int result = longest_non_decreasing_subsequence(sequence, n);
    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.

◄ 3-DP-Longest Common Subsequence

Jump to...

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

Started on	Friday, 23 August 2024, 2:23 PM
State	Finished
Completed on	Friday, 30 August 2024, 1:57 PM
Time taken	6 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

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>

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

int partition(int arr[], int low, int high)
{
    int pivot = arr[low];
    int i = low;
    int j = high;

    while (i < j)
    {
        while (arr[i] <= pivot && i <= high - 1) i++;
        while (arr[j] > pivot && j >= low + 1) j--;
        if (i < j)  swap(&arr[i], &arr[j]);
    }
    swap(&arr[low], &arr[j]);
    return j;
}

void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int partitionIndex = partition(arr, low, high);
        quickSort(arr, low, partitionIndex - 1);
        quickSort(arr, partitionIndex + 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);
    int sum=0;
    for(int i=0;i<n;++i)
    {
        sum+=i*arr[i];
    }
    printf("%d",sum);
```

```
52 }
```

	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.

◄ 3-G-Burger Problem

Jump to...

5-G-Product of Array elements-Minimum ►

Started on	Friday, 25 October 2024, 1:44 PM
State	Finished
Completed on	Friday, 25 October 2024, 1:47 PM
Time taken	3 mins 42 secs
Marks	1.00/1.00
Grade	**30.00** out of 30.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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>

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]) {
            i++;
        } else if (arr1[i] > arr2[j]) {
            j++;
        } else {

            printf("%d ", arr1[i]);
            i++;
            j++;
        }
    }
}

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);
        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.

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

Jump to...

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

Started on	Friday, 4 October 2024, 1:40 PM
State	Finished
Completed on	Friday, 4 October 2024, 1:40 PM
Time taken	14 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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 findPairRecursive(int arr[], int low, int high, int x) {
    if (low >= high) {
        return 0;
    }
    int left = low;
    int right = high;
    while (left < right) {
        int sum = arr[left] + arr[right];
        if (sum == x) {
            printf("%d\n", arr[left]);
            printf("%d\n", arr[right]);
            return 1;
        } else if (sum < x) {
            left++;
        } else {
            right--;
        }
    }
    return 0;
}
void checkPair(int arr[], int n, int x) {
    if (!findPairRecursive(arr, 0, n - 1, x)) {
        printf("No\n");
    }
}
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);
    checkPair(arr, n, x);
    return 0;
}

```

	Input	Expected	Got	
✔	4 2 4 8 10 14	4 10	4 10	✔

	Input	Expected	Got	
✔	5 2 4 6 8 10 100	No	No	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 3-Finding Floor Value

Jump to...

5-Implementation of Quick Sort ►

Started on	Friday, 23 August 2024, 2:31 PM
State	Finished
Completed on	Friday, 23 August 2024, 2:34 PM
Time taken	2 mins 53 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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 compare_ascending(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}
int compare_descending(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
}

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

    int* array_One = (int*)malloc(n * sizeof(int));
    int* array_Two = (int*)malloc(n * sizeof(int));

    if (array_One == NULL || array_Two == NULL) {
        perror("Failed to allocate memory");
        return 1;
    }
    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), compare_ascending);

    qsort(array_Two, n, sizeof(int), compare_descending);
    long long min_sum = 0;
    for (int i = 0; i < n; i++) {
        min_sum += (long long)array_One[i] * array_Two[i];
    }
    printf("%lld\n", min_sum);
    free(array_One);
    free(array_Two);

    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.

◄ 4-G-Array Sum max problem

Jump to...

1-Number of Zeros in a Given Array ►

Started on	Friday, 4 October 2024, 1:41 PM
State	Finished
Completed on	Friday, 4 October 2024, 1:41 PM
Time taken	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

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 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;
}
```

	Input	Expected	Got	
✔	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98	✔

	Input	Expected	Got	
✔	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.

◄ 4-Two Elements sum to x

Jump to...

1-DP-Playing with Numbers ►

Started on	Friday, 25 October 2024, 1:48 PM
State	Finished
Completed on	Friday, 25 October 2024, 2:04 PM
Time taken	16 mins 51 secs
Marks	1.00/1.00
Grade	**4.00** out of 4.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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 hasPairWithDifference(int arr[], int n, int k) {
    int left = 0, right = 1;

    while (right < n) {
        int diff = arr[right] - arr[left];

        if (diff == k && left != right) {
            return 1;
        } else if (diff < k) {
            right++;
        } else {
            left++;

            if (left == right) {
                right++;
            }
        }
    }

    return 0;
}

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

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

    int k;
    scanf("%d", &k);


    printf("%d\n", hasPairWithDifference(arr, n, k));

    return 0;
```

```
41 }
```

	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.

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

Jump to...

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

Started on	Friday, 25 October 2024, 2:05 PM
State	Finished
Completed on	Friday, 25 October 2024, 2:37 PM
Time taken	32 mins 34 secs
Marks	1.00/1.00
Grade	**4.00** out of 4.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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 hasPairWithDifference(int arr[], int n, int k) {
    int left = 0, right = 1;
    while (right < n) {
        int diff = arr[right] - arr[left];
        if (diff == k && left != right) {
            return 1;
        } else if (diff < k) {
            right++;
        } else {
            left++;

            if (left == right) {
                right++;
            }
        }
    }

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

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

Jump to...

Started on	Friday, 9 August 2024, 1:47 PM
State	Finished
Completed on	Friday, 9 August 2024, 2:03 PM
Time taken	16 mins 13 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

```
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>

void function (int n)
{
    int c=0;
    int i= 1;
    c++;
    int s =1;
    c++;
    while(s <= n)
    {
         c++;
         i++;
         c++;
         s += i;
         c++;
     }
    c++;
    printf("%d",c);
}

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

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Model exam DAA (B,D,E)

Jump to...

Problem 2: Finding Complexity using Counter method ►

Started on	Friday, 9 August 2024, 2:03 PM
State	Finished
Completed on	Friday, 9 August 2024, 2:15 PM
Time taken	12 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

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>

void func(int n) {
    int counter = 0;
    if (n == 1) {
        counter++;
       // printf("*");
        counter++;
    } else {
        for (int i = 1; i <= n; i++) {
            counter++;
            for (int j = 1; j <= n; j++) {
                counter++;
                //printf("*");
                counter++;
                //printf("*");
                counter++;
                break;
            }
            counter++;
        }
        counter++;
    }
    counter++;
    printf("%d\n", counter);
}

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

◄ Problem 1: Finding Complexity using Counter Method

Jump to...

Problem 3: Finding Complexity using Counter Method ►

Started on	Friday, 9 August 2024, 2:15 PM
State	Finished
Completed on	Friday, 9 August 2024, 2:17 PM
Time taken	2 mins 1 sec
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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:

```
#include<stdio.h>

void Factor(int num) {

     int c=0;
    for (int i = 1; i <= num;++i)
    {
        c++;
     if (num % i== 0)
        {
          c++;
          //printf("%d ", i);
        }
        c++;

     }
     c++;
     printf("%d",c);
  }
int main(){
    int n;
    scanf("%d",&n);
    Factor(n);
}
```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Problem 2: Finding Complexity using Counter method

Jump to...

Problem 4: Finding Complexity using Counter Method ►

Started on	Friday, 9 August 2024, 2:18 PM
State	Finished
Completed on	Friday, 9 August 2024, 2:43 PM
Time taken	24 mins 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

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:

```
#include<stdio.h>
int main()
{
    int n;
    int count=0;
    scanf("%d",&n);
    int c= 0;
    count++;
    for(int i=n/2; i<n; i++)
    {
        count++;
        for(int j=1; j<n; j = 2 * j)
        {
            count++;
            for(int k=1; k<n; k = k * 2)
            {
                count++;
                c++;
                count++;
            }
            count++;
        }
        count++;
    }
    count++;
    printf("%d",count);
}
```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Problem 3: Finding Complexity using Counter Method

Jump to...

Problem 5: Finding Complexity using counter method ►

Started on	Friday, 9 August 2024, 2:30 PM
State	Finished
Completed on	Friday, 9 August 2024, 2:30 PM
Time taken	24 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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:

```
#include<stdio.h>


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

    }
    c++;
    printf("%d",c);
}

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

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Problem 4: Finding Complexity using Counter Method

Jump to...

1-G-Coin Problem ►