Started on	Thursday, 5 September 2024, 8:50 AM
State	Finished
Completed on	Thursday, 5 September 2024, 9:08 AM
Time taken	17 mins 45 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 main()
{
    int n,c=0,i;
    scanf("%d",&n);
    int a[]={ 1, 2, 5, 10, 20, 50, 100, 500, 1000};
    for(i=8;i>=0;i--)
    {
        if(n>=a[i])
        {
            n=n-a[i];
            c++;
        }
    }
    printf("%d",c);
}
```

	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ 6-Implementation of Quick Sort

Jump to...

2-G-Cookies Problem ►

Started on	Thursday, 3 October 2024, 8:45 AM
State	Finished
Completed on	Thursday, 3 October 2024, 8:46 AM
Time taken	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

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 %)

```
#include <stdio.h>
int majorityElement(int* n
int candidate = nums[0];
int count = 1;
for (int i = 1; i < numsSi
if (nums[i] == candidate)
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 majority = majorityEle
printf("%d\n", majority);
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	Thursday, 3 October 2024, 8:46 AM
State	Finished
Completed on	Thursday, 3 October 2024, 8:47 AM
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:

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 left = 0, right = n - 1, floorIndex = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x) {
            return arr[mid];
        } else if (arr[mid] < x) {
            floorIndex = mid;
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return (floorIndex != -1) ? arr[floorIndex] : -1;
}
int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int x;
    scanf("%d", &x);
    int floorValue = findFloor(arr, n, x);
    printf("%d\n", floorValue);
    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	Thursday, 12 September 2024, 8:41 AM
State	Finished
Completed on	Thursday, 12 September 2024, 9:18 AM
Time taken	37 mins 42 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 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 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);
    int max_sum = 0;
    for (int i = 0; i < n; i++) {
        max_sum += arr[i] * i;
    }
    printf("%d\n", max_sum);
    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	✔

	Input	Expected	Got	
✔	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	Thursday, 12 September 2024, 9:19 AM
State	Finished
Completed on	Thursday, 12 September 2024, 9:19 AM
Time taken	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

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_asc(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}
int compare_desc(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));
    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_asc);
    qsort(array_Two, n, sizeof(int), compare_desc);
    int min_sum = 0;
    for (int i = 0; i < n; i++) {
        min_sum += array_One[i] * array_Two[i];
    }
    printf("%d\n", min_sum);
    free(array_One);
    free(array_Two);
    return 0;
}
```

	Input	Expected	Got	
✔	3 1 2 3 4 5 6	28	28	✔

	Input	Expected	Got	
✔	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-DP-Playing with Numbers ►