2

1

Sample Output 1

1

Explanation 1

- The first and last elements are equal to 1.
- Using zero based indexing, arr[1]=2 is the pivot between the two subarrays.
- The index of the pivot is 1.

**Answer:** (penalty regime: 0 %)

Reset answer

```c
/*
 * Complete the 'balancedSum' function be
 *
 * The function is expected to return an
 * The function accepts INTEGER_ARRAY arr
 */

int balancedSum(int arr_count, int* arr)
{
    int totalSum=0,leftSum=0;
    for(int i=0;i<arr_count;i++)
    {
        totalSum+=arr[i];
    }
    for(int i=0;i<arr_count;i++)
    {
        totalSum-=arr[i];
        if(leftSum==totalSum)
        {
            return i;
        }
        leftSum+=arr[i];
    }
    return 1;

}
```

| | Test | Expected |
|---|---|---|
| ✓ | `int arr[] = {1,2,3,3};`<br>`printf("%d", balancedSum(4, arr))` | 2 |

Passed all tests! ✓

Question **2**

Correct

Flag question

Calculate the sum of an array of integers.

Example

numbers = [3, 13, 4, 11, 9]

The sum is 3 + 13 + 4 + 11 + 9 = 40.

Function Description

Complete the function arraySum in the editor below.

arraySum has the following parameter(s):

int numbers[n]: an array of integers

Returns

int: integer sum of the numbers array

12 → numbers = [12, 12]

12

Sample Output 1

24

Explanation 1

12 + 12 = 24.

**Answer:** (penalty regime: 0 %)

Reset answer

```
/*
 * Complete the 'arraySum' function below
 *
 * The function is expected to return an
 * The function accepts INTEGER_ARRAY num
 */

int arraySum(int numbers_count, int *numb
{
    int sum=0;
    for(int i=0;i<numbers_count;i++)
    {
        sum+=numbers[i];
    }
    return sum;

}
```

| | Test | Expected | Go |
|---|---|---|---|
| ✓ | `int arr[] = {1,2,3,4,5}; printf("%d", arraySum(5, arr))` | 15 | 15 |

Passed all tests! ✓

Question **3**

Correct

Flag question

Given an array of n integers, rearrange them so that the sum of the absolute differences of all adjacent elements is minimized. Then, compute the sum of those absolute differences. Example n = 5 arr = [1, 3, 3, 2, 4] If the list is rearranged as arr' = [1, 2, 3, 3, 4], the absolute differences are |1 - 2| = 1, |2 - 3| = 1, |3 - 3| = 0, |3 - 4| = 1. The sum of those differences is 1 + 1 + 0 + 1 = 3. Function Description Complete the function minDiff in the editor below. minDiff has the following parameter: arr: an integer array Returns: int: the sum of the absolute differences of adjacent elements Constraints 2 ≤ n ≤105 0 ≤ arr[i] ≤ 109, where 0 ≤ i < n Input Format For Custom Testing The first line of input contains an integer, n, the size of arr. Each of the following n lines contains an integer that describes arr[i] (where 0 ≤ i < n) . Sample Case 0 Sample Input For Custom Testing STDIN Function ----- -------- 5 → arr[] size n = 5 5 → arr[] = [5, 1, 3, 7, 3] 1 3 7 3 Sample Output 6 Explanation n = 5 arr = [5, 1, 3, 7, 3] If arr is rearranged as arr' = [1, 3, 3, 5, 7], the differences are minimized. The final answer is |1 - 3| + |3 - 3| + |3 - 5| + |5 - 7| = 6. Sample Case 1 Sample Input For Custom Testing STDIN Function ----- -------- 2 → arr[] size n = 2 3 → arr[] = [3, 2] 2 Sample Output 1 Explanation n = 2 arr = [3, 2] There is no need to rearrange because there are only two elements. The final answer is |3 - 2| = 1.

**Answer:** (penalty regime: 0 %)

Reset answer

```
/*
 * Complete the 'minDiff' function below.
 *
 * The function is expected to return an
 * The function accepts INTEGER_ARRAY arr
 */
```

```
 */

int arraySum(int numbers_count, int *numb
{
    int sum=0;
    for(int i=0;i<numbers_count;i++)
    {
        sum+=numbers[i];
    }
    return sum;

}
```

| | Test | Expected | Go |
|---|---|---|---|
| ✓ | `int arr[] = {1,2,3,4,5}; printf("%d", arraySum(5, arr))` | 15 | 15 |

Passed all tests! ✓

Question 3

Correct

Flag question

Given an array of n integers, rearrange them so that the sum of the absolute differences of all adjacent elements is minimized. Then, compute the sum of those absolute differences. Example n = 5 arr = [1, 3, 3, 2, 4] If the list is rearranged as arr' = [1, 2, 3, 3, 4], the absolute differences are |1 - 2| = 1, |2 - 3| = 1, |3 - 3| = 0, |3 - 4| = 1. The sum of those differences is 1 + 1 + 0 + 1 = 3. Function Description Complete the function minDiff in the editor below. minDiff has the following parameter: arr: an integer array Returns: int: the sum of the absolute differences of adjacent elements Constraints 2 ≤ n ≤105 0 ≤ arr[i] ≤ 109, where 0 ≤ i < n Input Format For Custom Testing The first line of input contains an integer, n, the size of arr. Each of the following n lines contains an integer that describes arr[i] (where 0 ≤ i < n) . Sample Case 0 Sample Input For Custom Testing STDIN Function ----- -------- 5 → arr[] size n = 5 5 → arr[] = [5, 1, 3, 7, 3] 1 3 7 3 Sample Output 6 Explanation n = 5 arr = [5, 1, 3, 7, 3] If arr is rearranged as arr' = [1, 3, 3, 5, 7], the differences are minimized. The final answer is |1 - 3| + |3 - 3| + |3 - 5| + |5 - 7| = 6. Sample Case 1 Sample Input For Custom Testing STDIN Function ----- -------- 2 → arr[] size n = 2 3 → arr[] = [3, 2] 2 Sample Output 1 Explanation n = 2 arr = [3, 2] There is no need to rearrange because there are only two elements. The final answer is |3 - 2| = 1.

**Answer:** (penalty regime: 0 %)

Reset answer

```
/*
 * Complete the 'minDiff' function below.
 *
 * The function is expected to return an
 * The function accepts INTEGER_ARRAY arr
 */

int compare(const void *a,const void*b)
{
    return(*(int*)a-*(int*)b);

}
int minDiff(int arr_count,int*arr)
{
    qsort(arr,arr_count,sizeof(int),compa
    int min_sum=0;
    for(int i=1;i<arr_count;i++)
    {
        min_sum+=abs(arr[i]-arr[i-1]);
    }
    return min_sum;
}
```

| | Test | Expected | Got |
|---|---|---|---|
| ✓ | `int arr[] = {5, 1, 3, 7, 3}; printf("%d", minDiff(5, arr))` | 6 | 6 |

Passed all tests! ✓