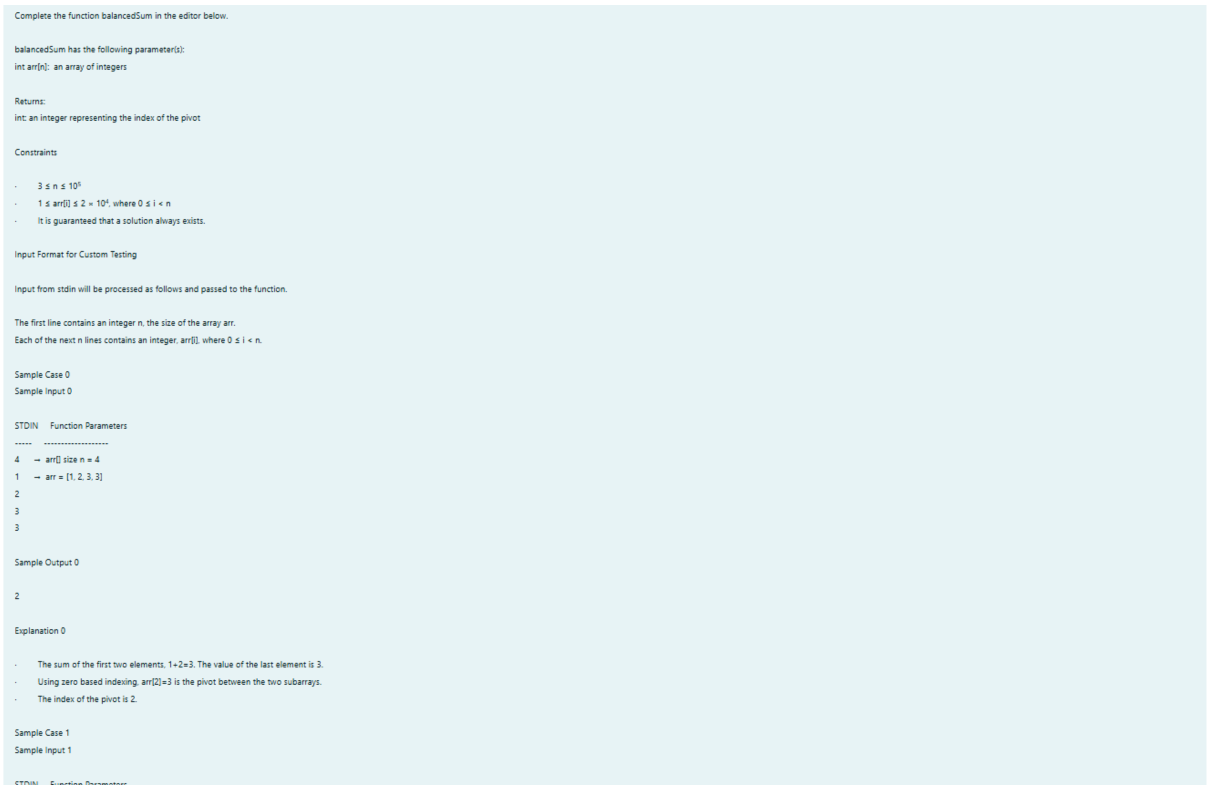

Complete the function balancedSum in the editor below.

balancedSum has the following parameter(s):
int arr[n]: an array of integers

Returns:
int: an integer representing the index of the pivot

Constraints

- $3 \le n \le 10^5$
- $1 \le arr[i] \le 2 \times 10^4$, where $0 \le i < n$
- It is guaranteed that a solution always exists.

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n, the size of the array arr.
Each of the next n lines contains an integer, arr[i], where $0 \le i < n$.

Sample Case 0
Sample Input 0

```
STDIN     Function Parameters
-----     -------------------
4      →  arr[] size n = 4
1      →  arr = [1, 2, 3, 3]
2
3
3
```

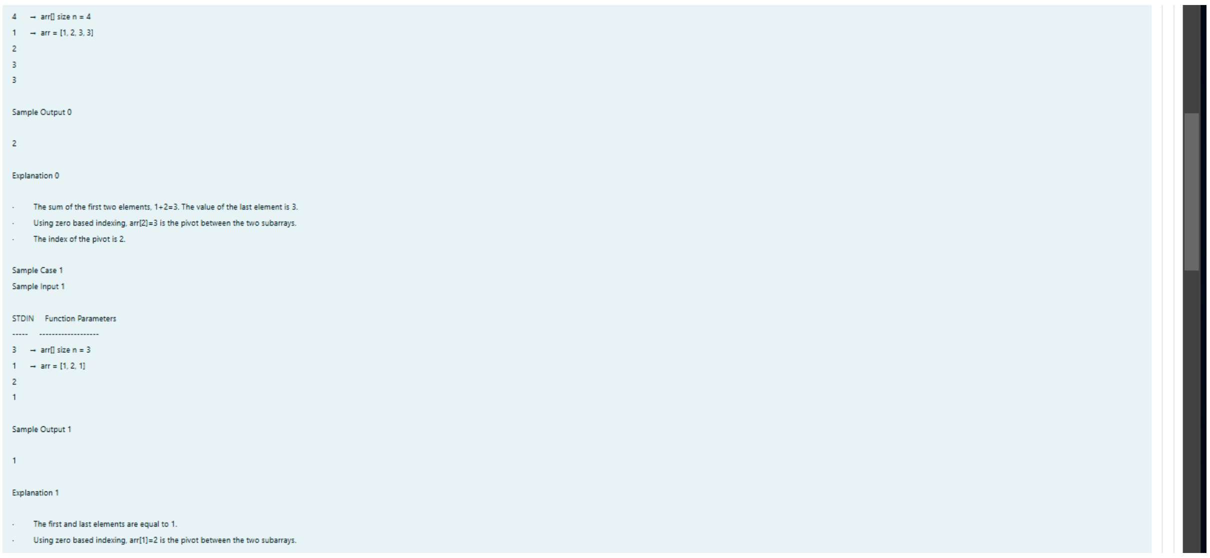
Sample Output 0

```
2
```

Explanation 0

- The sum of the first two elements, 1+2=3. The value of the last element is 3.
- Using zero based indexing, arr[2]=3 is the pivot between the two subarrays.
- The index of the pivot is 2.

Sample Case 1
Sample Input 1



```
4      →  arr[] size n = 4
1      →  arr = [1, 2, 3, 3]
2
3
3
```

Sample Output 0

```
2
```

Explanation 0

- The sum of the first two elements, 1+2=3. The value of the last element is 3.
- Using zero based indexing, arr[2]=3 is the pivot between the two subarrays.
- The index of the pivot is 2.

Sample Case 1
Sample Input 1

```
STDIN     Function Parameters
-----     -------------------
3      →  arr[] size n = 3
1      →  arr = [1, 2, 1]
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.

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 below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER_ARRAY arr as parameter.
 */

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

}
```

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

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

Constraints

$1 \le n \le 10^4$
$1 \le numbers[i] \le 10^4$

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n, the size of the array numbers.
Each of the next n lines contains an integer numbers[i] where $0 \le i < n$.

Sample Case 0
Sample Input 0

```
STDIN     Function
-----     --------
5    →  numbers[] size n = 5
1    →  numbers = [1, 2, 3, 4, 5]
2
3
4
5
```

Sample Output 0

15

Explanation 0

1 + 2 + 3 + 4 + 5 = 15.

Sample Case 1

Sample Input 1

```
STDIN     Function
-----     --------
2    →  numbers[] size n = 2
12   →  numbers = [12, 12]
12
```

Sample Output 1

24

Explanation 1

12 + 12 = 24.

Answer: (penalty regime: 0 %)

Reset answer

```c
/*
 * Complete the 'arraySum' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER_ARRAY numbers as parameter.
 */

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

}

```

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

Passed all tests! ✓

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

```c
/*
 * Complete the 'minDiff' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER_ARRAY arr as parameter.
 */
#include<stdlib.h>
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),compare);
    int totaldiff=0;
    for(int i=1;i<arr_count;i++)
    {
        totaldiff+=abs(arr[i]-arr[i-1]);
    }
    return totaldiff;

}

```

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

Passed all tests! ✓