PL-2020-C-Missing Numbers

locked

Numeros the Artist had two lists that were permutations of one another. He was very proud. Unfortunately, while transporting them from one exhibition to another, some numbers were lost out of the first list. Can you find the missing numbers?

As an example, the array with some numbers missing, arr = [7, 2, 5, 3, 5, 3]. The original array of numbers brr = [7, 2, 5, 4, 6, 3, 5, 3]. The numbers missing are [4, 6].

Notes

• If a number occurs multiple times in the lists, you must ensure that the frequency of that number in both lists is the same. If that is not the case, then it is also a missing number. • You have to print all the missing numbers in ascending order. • Print each missing number once, even if it is missing multiple times. • The difference between maximum and minimum number in the second list is less than or equal to 100.

Complete the code in the editor below. It should return a sorted array of missing numbers.

It has the following: • arr: the array with missing numbers • brr: the original array of numbers

Input Format

There will be four lines of input: n - the size of the first list, arr The next line contains n space-separated integers arr[i] m - the size of the second list, brr The next line contains m space-separated integers brr[i]

Constraints

• 1 ≤ n, m ≤ 2 x 10^5 • n ≤ m • 1 ≤ brr[i] ≤ 2 x 10^4 • Xmax – Xmin < 101

Output Format

Output the missing numbers in ascending order.

Sample Input 0

```
10
203 204 205 206 207 208 203 204 205 206
13
203 204 204 205 206 207 205 208 203 206 205 206 204
```

Sample Output 0

```
204 205 206
```

Explanation 0

204 is present in both arrays. Its frequency in arr is 2, while its frequency in brr is 3. Similarly, 205 and 206 occur twice in arr, but three times in brr. The rest of the numbers have the same frequencies in both lists.

Submissions: 658
Max Score: 100

Difficulty: Medium

Rate This Challenge:

More

C

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

#define read_int(x) scanf("%d", &x)
#define RANGE 100
#define BASE_IDX (RANGE + 1)
#define COUNT_SIZE (RANGE * 2)
int main() {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    int m, n;
    int base, k, i;
    int count[COUNT_SIZE];
    memset(count, 0, COUNT_SIZE * sizeof(int));
    read_int(m);
    read_int(k);
    base = k;
    count[BASE_IDX] = 1;
    for (i = 1; i < m; i++) {
        read_int(k);
        count[BASE_IDX + (k - base)]++;
    }
    read_int(n);
    for (i = 0; i < n; i++) {
        read_int(k);
        count[BASE_IDX + (k - base)]--;
    }

    for (i = 0; i < COUNT_SIZE; i++) {
        if (count[i] < 0) printf("%d ", base + (i - BASE_IDX));
    }
    printf("\n");
    return 0;
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code