```cpp
//{ Driver Code Starts
#include <bits/stdc++.h>
using namespace std;

// } Driver Code Ends

class Solution
{
public:
    // Function to swap elements:
    void swap(int *a, int *b)
    {
        int t = *a;
        *a = *b;
        *b = t;
    }

    // Function to rearrange array (find the partition point)
    int partition(int array[], int low, int high)
    {

        // Select the rightmost element as pivot
        int pivot = array[high];

        // Pointer for greater element
        int i = (low - 1);

        /*
        Traverse each element of the array and
        compare them with the pivot:
        */
        for (int j = low; j < high; j++)
        {
            if (array[j] <= pivot)
            {

                /*
                If an element smaller than the pivot is found,
                swap it with the greater element pointed by i:
                */
                i++;

                // Swap element at i with element at j:
                swap(&array[i], &array[j]);
            }
        }

        // Swap pivot with the greater element at i:
        swap(&array[i + 1], &array[high]);

        // Return the partition point:
        return (i + 1);
    }
    void quickSort(int array[], int low, int high)
    {
        if (low < high)
        {
            /*
            Find the pivot element such that:
            elements smaller than pivot are to the left of pivot,
            and elements greater than pivot are to the right of
pivot:
            */
            int pi = partition(array, low, high);

            // Recursive call on the left of pivot:
            quickSort(array, low, pi - 1);

            // Recursive call on the right of pivot:
            quickSort(array, pi + 1, high);
        }
    }

    long long int minValue(int a[], int b[], int n)
    {
        // Your code goes here

        /*
        Sort arrays a[] and b[] using the "quicksort" algorithm
above
        into increasing arrays:
        */
        quickSort(a, 0, n - 1);
        quickSort(b, 0, n - 1);

        /*
        Initialise a 64 bit integer counter as "minsum",
        where the 'long long' (64 bit size) is needed for
        the larger test cases:
        */
        long long int minsum = 0;

        /*
        Take the ith element a[i] of the sorted a[] array, and
        multiply it with the (n-i-1) element b[n-n-i] of the
        sorted b[n-i-1] array (as they are both sorted into
        an increasing order):
        */
        for (int i = 0; i < n; i++)
        {
            minsum += a[i] * b[n - i - 1];
        }

        return minsum;
    }
};
```

```
//{ Driver Code Starts.
int main()
{
    int t;
    cin >> t;
    while (t--)
    {
        int n, i;
        cin >> n;
        int a[n], b[n];
        for (i = 0; i < n; i++)
            cin >> a[i];
        for (i = 0; i < n; i++)
            cin >> b[i];
        Solution ob;
        cout << ob.minValue(a, b, n) << endl;
    }

    return 0;
}
// } Driver Code Ends
```

Problems
Courses
Get Hired
Contests
</>POTD
Practice
5
</> Problem
Editorial
Submissions
Comments
C++ (g++ 5.4)
Custom Input
Compile & Run
Submit

Average Time: 20m
Your Time: 23m

Output Window

Compilation Results | Custom Input

Problem Solved Successfully

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed:
150 /150

Your Total Score:
5

Total Time Taken:
0.55

Correct Submission Count:
4

Attempts No.:
15

```
    long long int minValue(int a[], int b[], int n)
    {
        // Your code goes here

        /*
        Sort arrays a[] and b[] using the "quicksort" algorithm above
        into increasing arrays:
        */
        quickSort(a, 0, n - 1);
        quickSort(b, 0, n - 1);

        /*
        Initialise a 64 bit integer counter as "minsum",
        where the 'long long' (64 bit size) is needed for
        the larger test cases:
        */
        long long int minsum = 0;

        /*
        Take the ith element a[i] of the sorted a[] array, and
        multiply it with the (n-i-1) element b[n-n-i] of the
        sorted b[n-i-1] array (as they are both sorted into
        an increasing order):
        */
        for (int i = 0; i < n; i++)
        {
            minsum += a[i] * b[n - i - 1];
        }

        return minsum;
    }
};
```