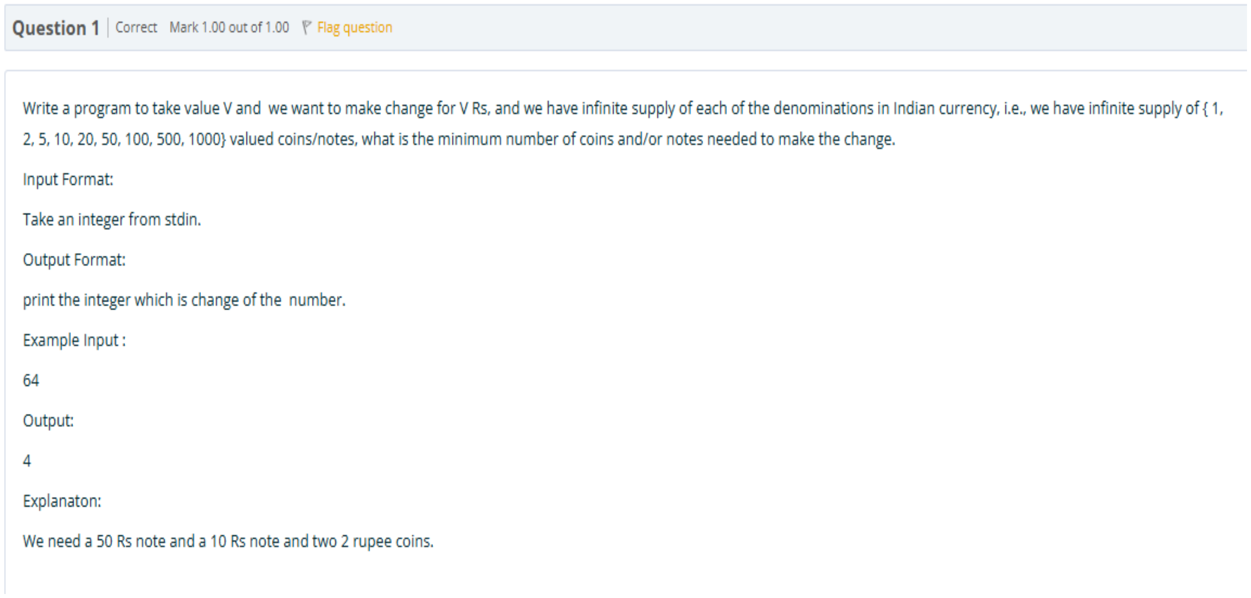
Greedy Algorithms

1-G-Coin Problem:



Question 1 | Correct Mark 1.00 out of 1.00 Flag question

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 :

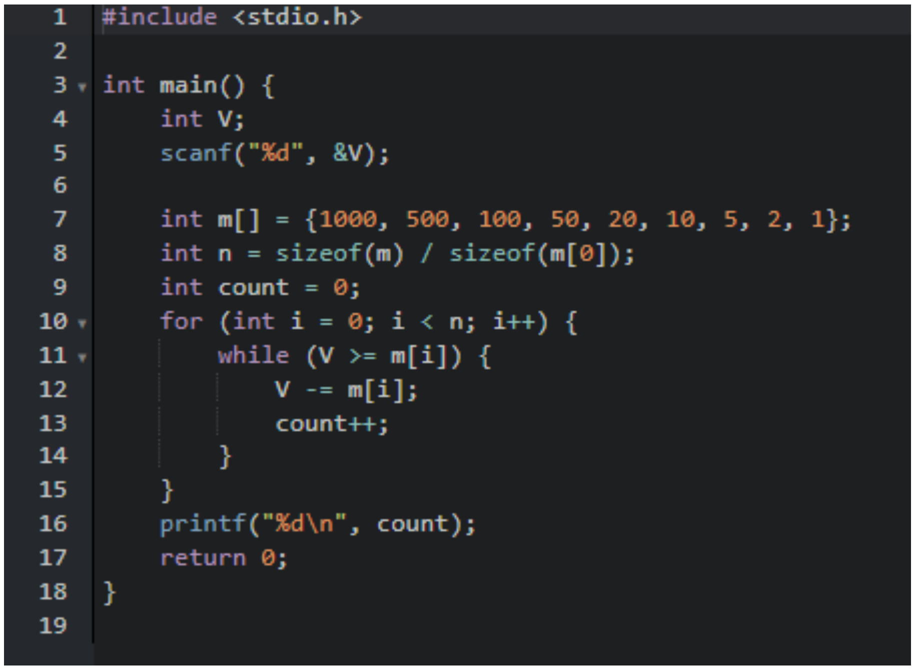
64

Output:

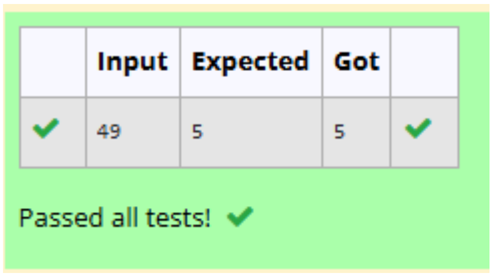
4

Explanaton:

We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.



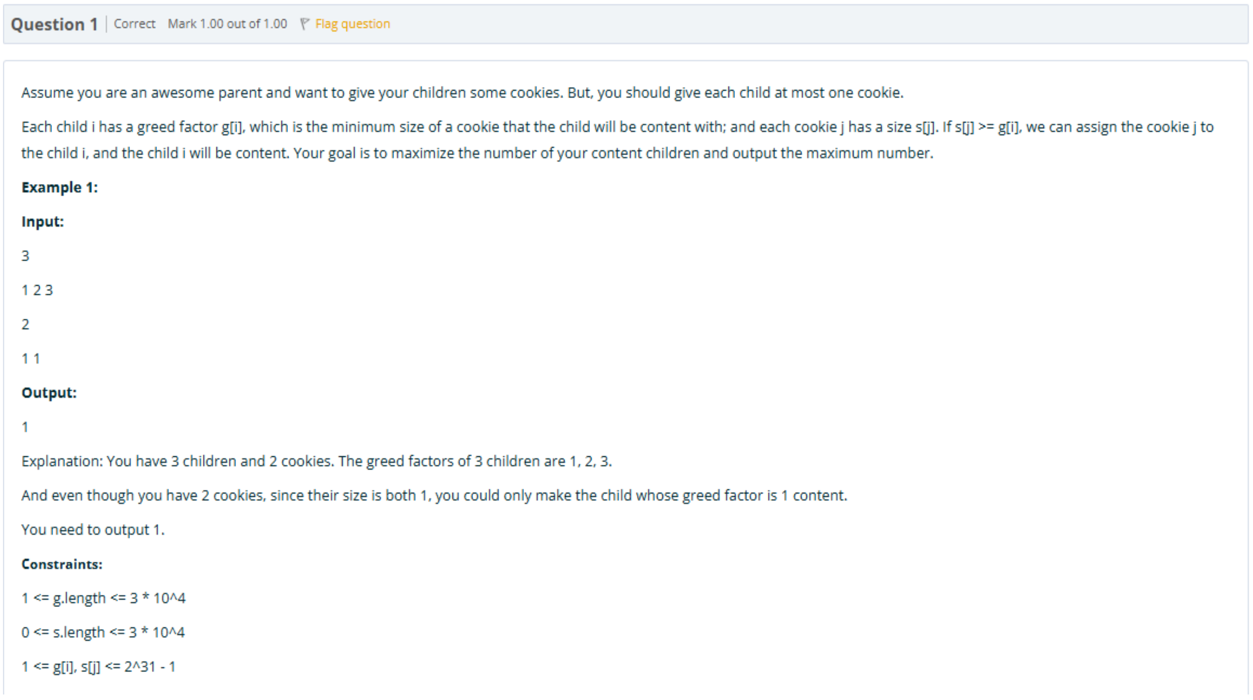
```c
#include <stdio.h>

int main() {
    int V;
    scanf("%d", &V);

    int m[] = {1000, 500, 100, 50, 20, 10, 5, 2, 1};
    int n = sizeof(m) / sizeof(m[0]);
    int count = 0;
    for (int i = 0; i < n; i++) {
        while (V >= m[i]) {
            V -= m[i];
            count++;
        }
    }
    printf("%d\n", count);
    return 0;
}

```



	Input	Expected	Got	
✓	49	5	5	✓

Passed all tests! ✓

2-G-Cookies Problem:



Question 1 Correct Mark 1.00 out of 1.00 Flag question

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.

Each child i has a greed factor g[i], which is the minimum size of a cookie that the child will be content with; and each cookie j has a size s[j]. If s[j] >= g[i], we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and output the maximum number.

Example 1:

Input:

3

1 2 3

2

1 1

Output:

1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

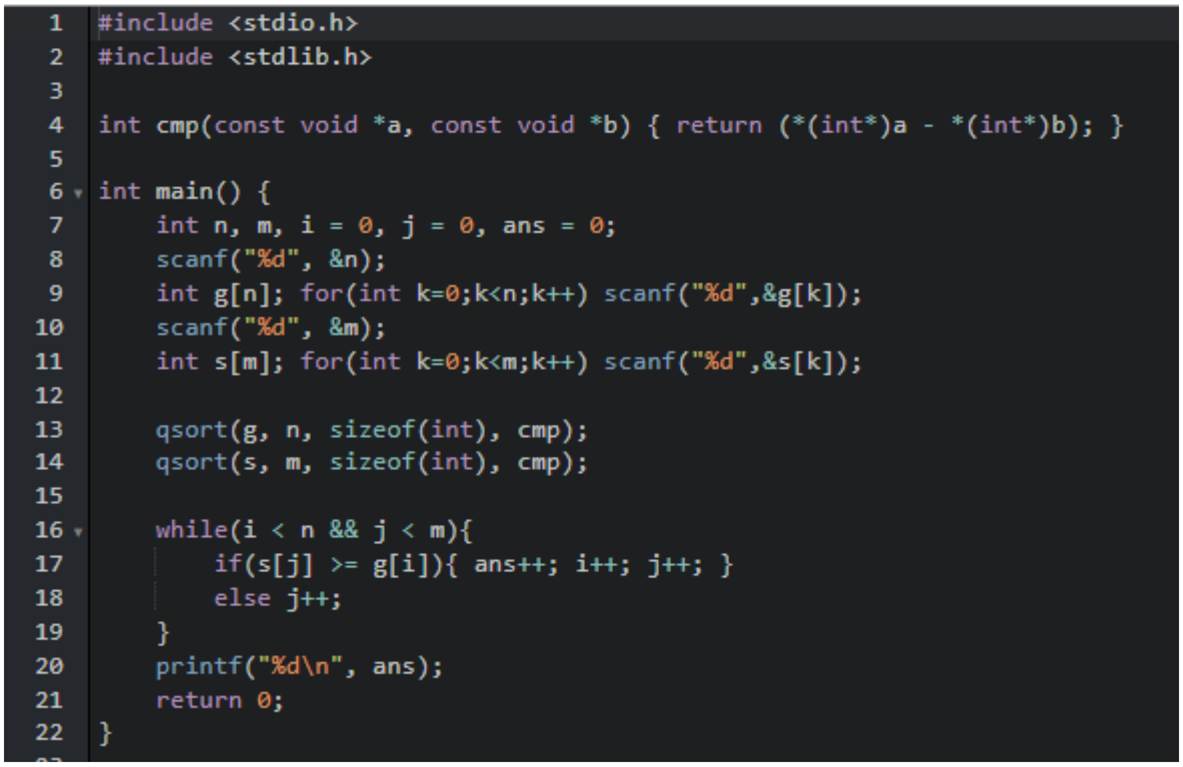
You need to output 1.

Constraints:

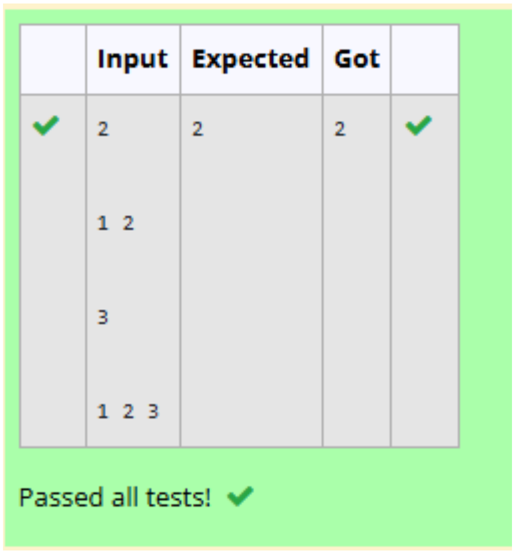
1 <= g.length <= 3 * 10^4

0 <= s.length <= 3 * 10^4

1 <= g[i], s[j] <= 2^31 - 1



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

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

int main() {
    int n, m, i = 0, j = 0, ans = 0;
    scanf("%d", &n);
    int g[n]; for(int k=0;k<n;k++) scanf("%d",&g[k]);
    scanf("%d", &m);
    int s[m]; for(int k=0;k<m;k++) scanf("%d",&s[k]);

    qsort(g, n, sizeof(int), cmp);
    qsort(s, m, sizeof(int), cmp);

    while(i < n && j < m){
        if(s[j] >= g[i]){ ans++; i++; j++; }
        else j++;
    }
    printf("%d\n", ans);
    return 0;
}
```



	Input	Expected	Got	
✓	2 1 2 3 1 2 3	2	2	✓

Passed all tests! ✓

4-G-Array Sum max problem:

Question 1 Correct Mark 1.00 out of 1.00 Flag question

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:

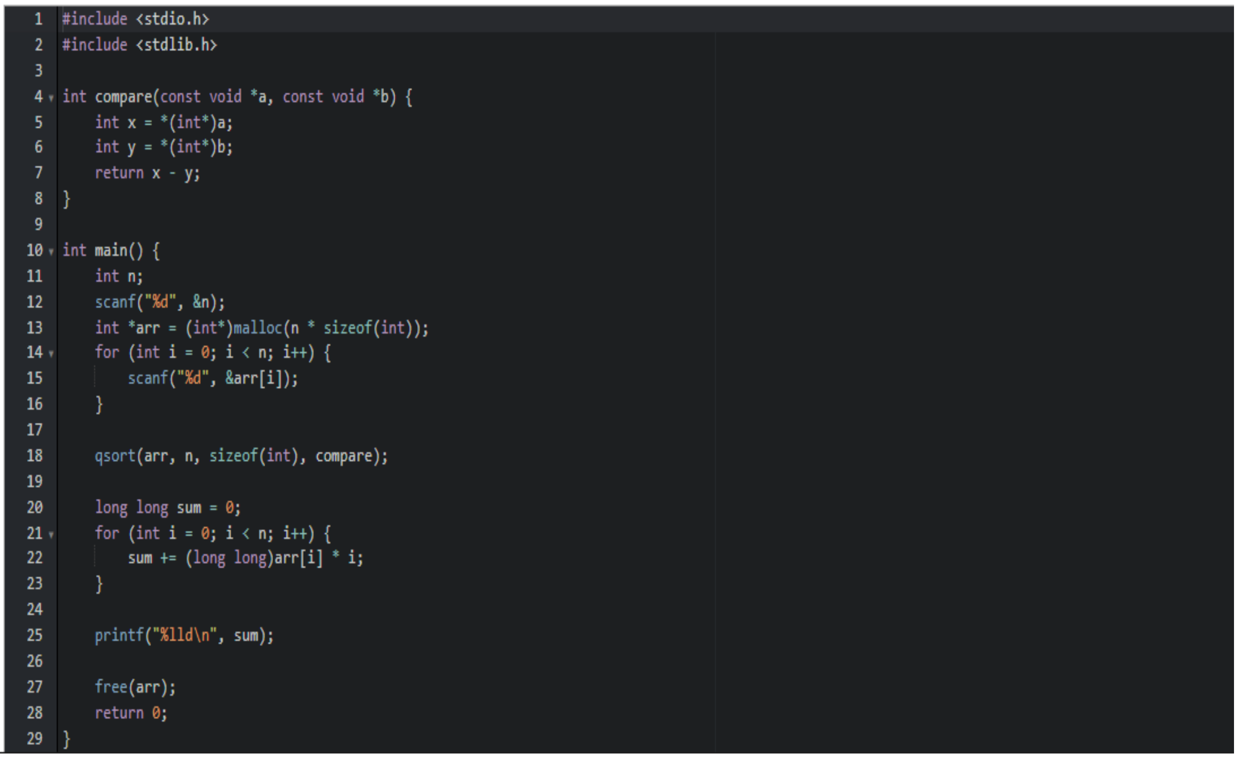
5

2 5 3 4 0

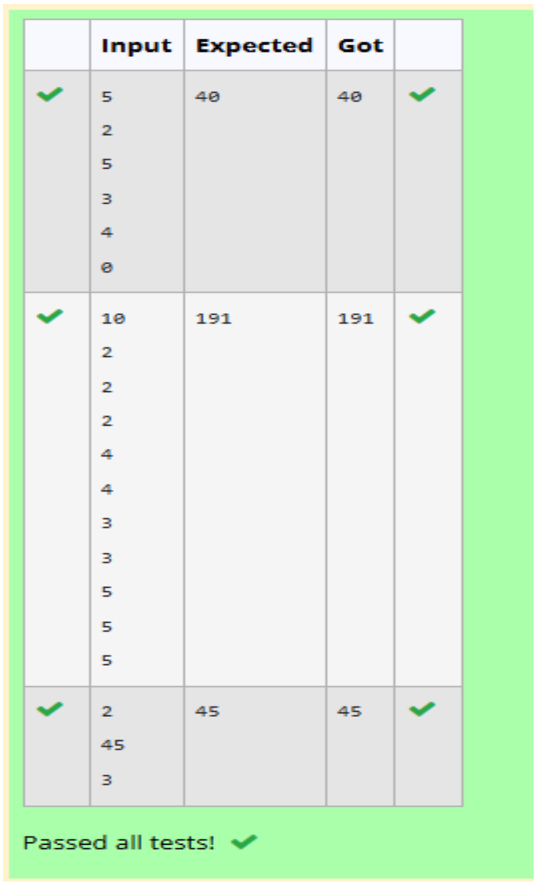
Sample output:

40

Answer: (penalty regime: 0 %)



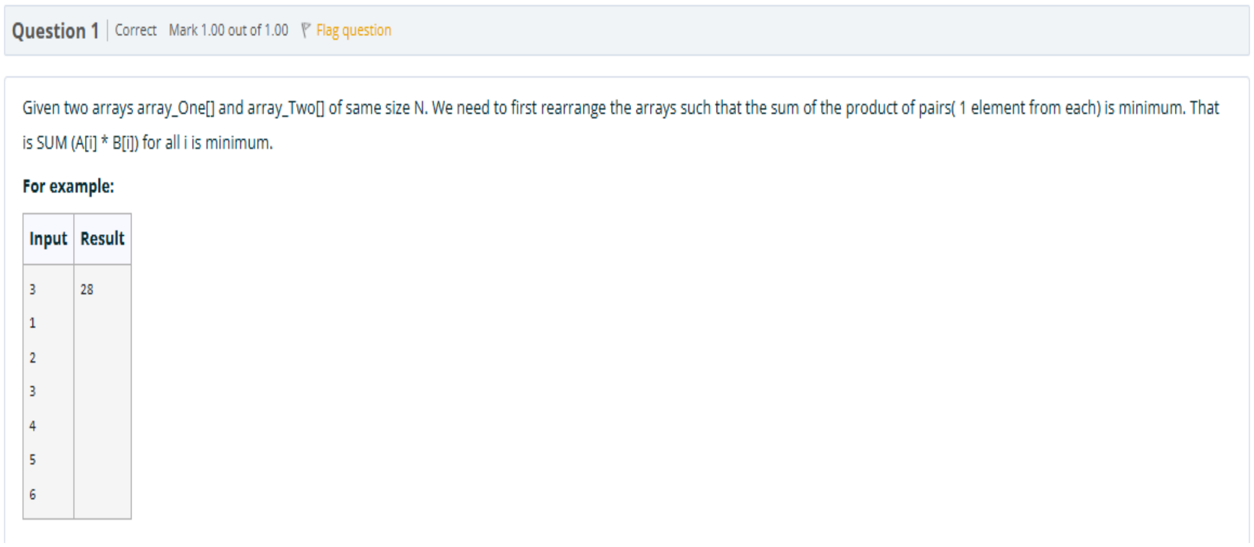
```c
#include <stdio.h>
#include <stdlib.h>

int compare(const void *a, const void *b) {
    int x = *(int*)a;
    int y = *(int*)b;
    return x - y;
}

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);

    long long sum = 0;
    for (int i = 0; i < n; i++) {
        sum += (long long)arr[i] * i;
    }

    printf("%lld\n", 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	✓
✓	2 45 3	45	45	✓

Passed all tests! ✓

5-G-Product of Array elements-Minimum:

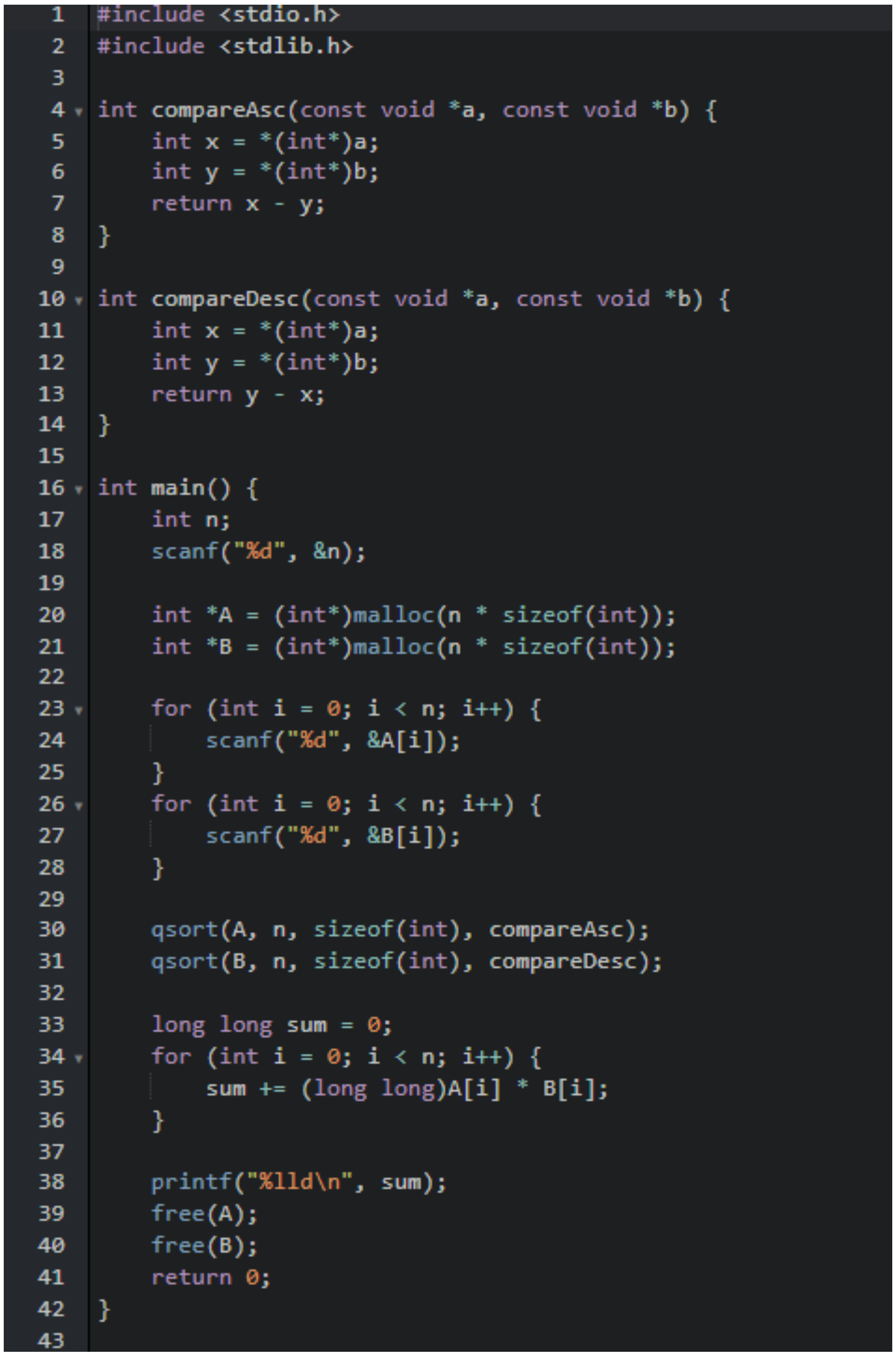


Question 1 Correct Mark 1.00 out of 1.00 Flag question

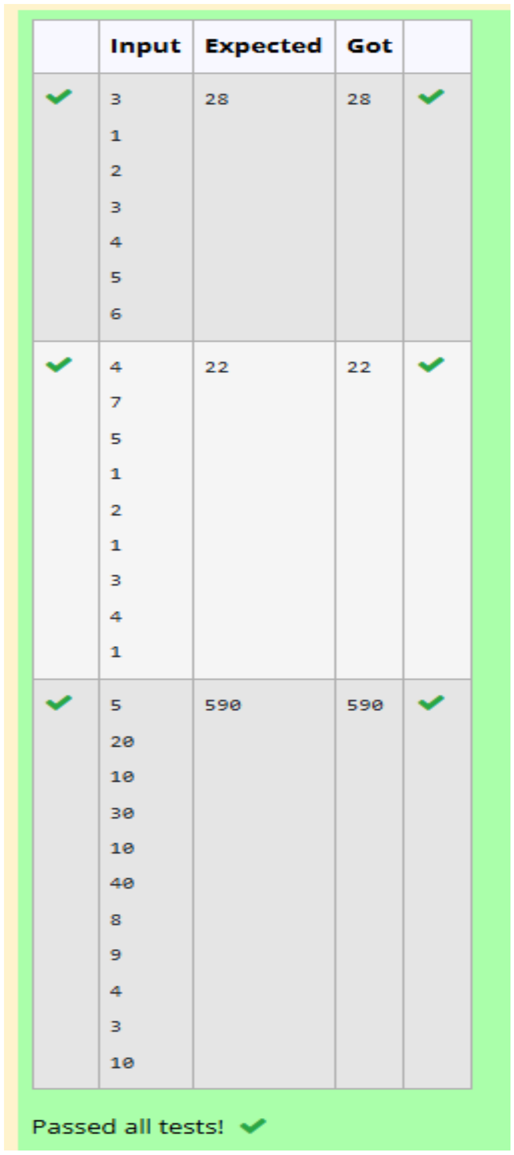
Given two arrays array_One[] and array_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs(1 element from each) is minimum. That is SUM (A[i] * B[i]) for all i is minimum.

For example:

Input	Result
3 1 2 3 4 5 6	28



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

int compareAsc(const void *a, const void *b) {
    int x = *(int*)a;
    int y = *(int*)b;
    return x - y;
}

int compareDesc(const void *a, const void *b) {
    int x = *(int*)a;
    int y = *(int*)b;
    return y - x;
}

int main() {
    int n;
    scanf("%d", &n);

    int *A = (int*)malloc(n * sizeof(int));
    int *B = (int*)malloc(n * sizeof(int));

    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }
    for (int i = 0; i < n; i++) {
        scanf("%d", &B[i]);
    }

    qsort(A, n, sizeof(int), compareAsc);
    qsort(B, n, sizeof(int), compareDesc);

    long long sum = 0;
    for (int i = 0; i < n; i++) {
        sum += (long long)A[i] * B[i];
    }

    printf("%lld\n", sum);
    free(A);
    free(B);
    return 0;
}

```



	Input	Expected	Got	
✓	3 1 2 3 4 5 6	28	28	✓
✓	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! ✓