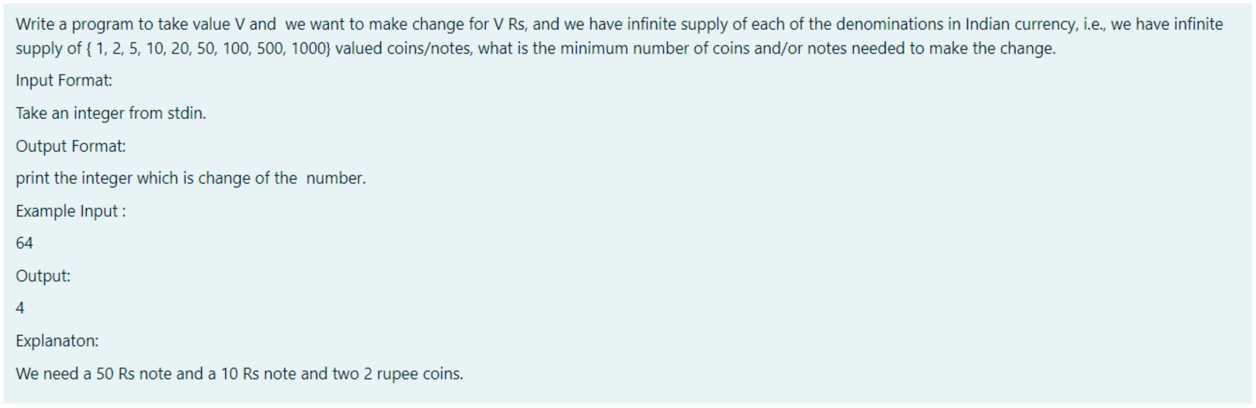
GREEDY ALGORITHM

1.



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:

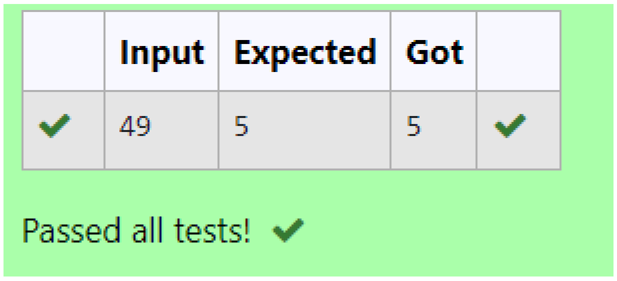
We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

Algorithm:

1. Start.
2. Input integer n.
3. Initialize count = 0, and array arr = {1, 2, 5, 10, 20, 50, 100, 500, 1000}.
4. While n != 0:
5. Set largest = 0.
6. For each i from 0 to 8:
   a. If n >= arr[i], check if arr[i] > largest, and if true, update largest.
7. Subtract largest from n.
8. Increment count by 1.
9. Output value of count.
10. End.

Program:

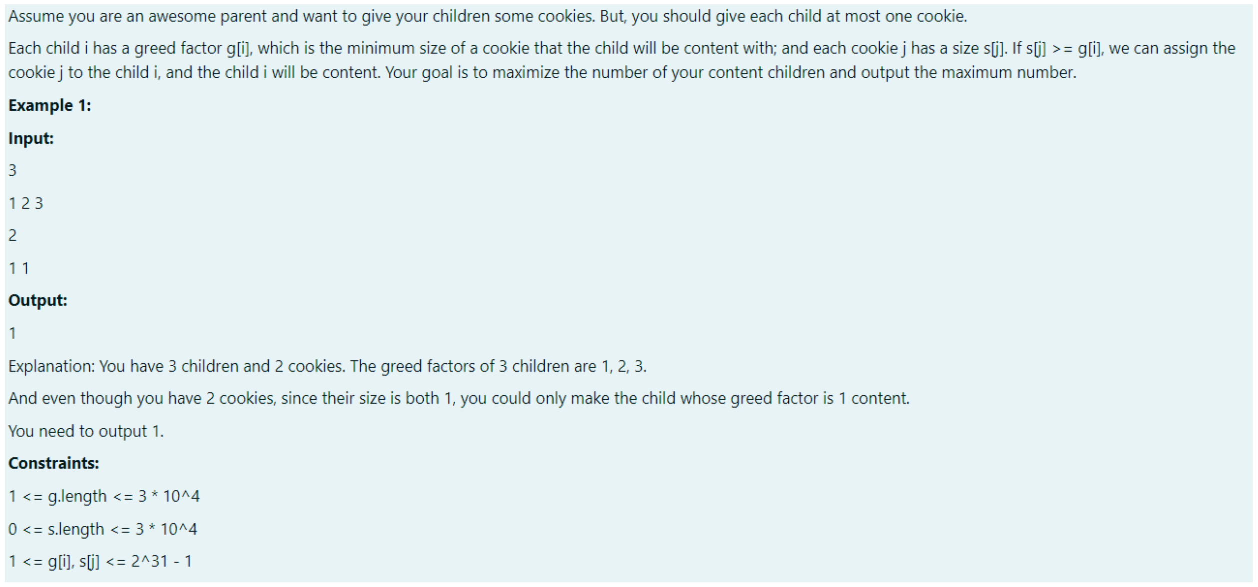
```
#include<stdio.h>
int main(){
   int n;
   scanf("%d",&n);
   int count=0;
   int arr[9]={1,2,5,10,20,50,100,500,1000};
   while(n!=0){
      int largest=0;
      for(int i=0;i<9;i++){
         if(n>=arr[i]){
            if(arr[i]>largest){
               largest=arr[i];
            }

         }
      }
      n-=largest;
      count++;
   }
   printf("%d",count);

}
```



	Input	Expected	Got	
✓	49	5	5	✓

Passed all tests! ✓

2.



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

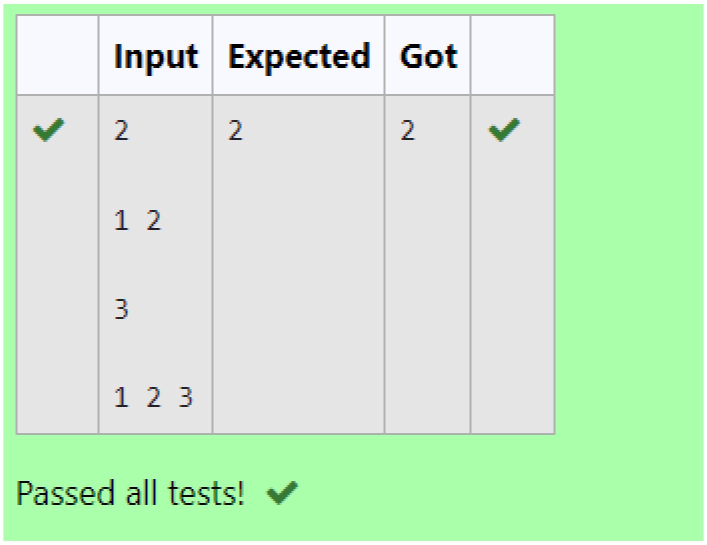
Algorithm:

1. Start
2. Input integer child
3. Input array greed[child]
4. Input integer cookies
5. Input array ckgreed[cookies]
6. Initialize count = 0
7. For each child i = 0 to child - 1
8. For each cookie j = 0 to cookies - 1If ckgreed[j] >= greed[i], increment count, and break
9. Output value of count
10. End

Program:

```
#include<stdio.h>
int main(){
  int child;
  scanf("%d",&child);
  int greed[child];
  for(int i=0;i<child;i++){
    scanf("%d",&greed[i]);
  }
  int cookies;
  scanf("%d",&cookies);
  int ckgreed[cookies];
```

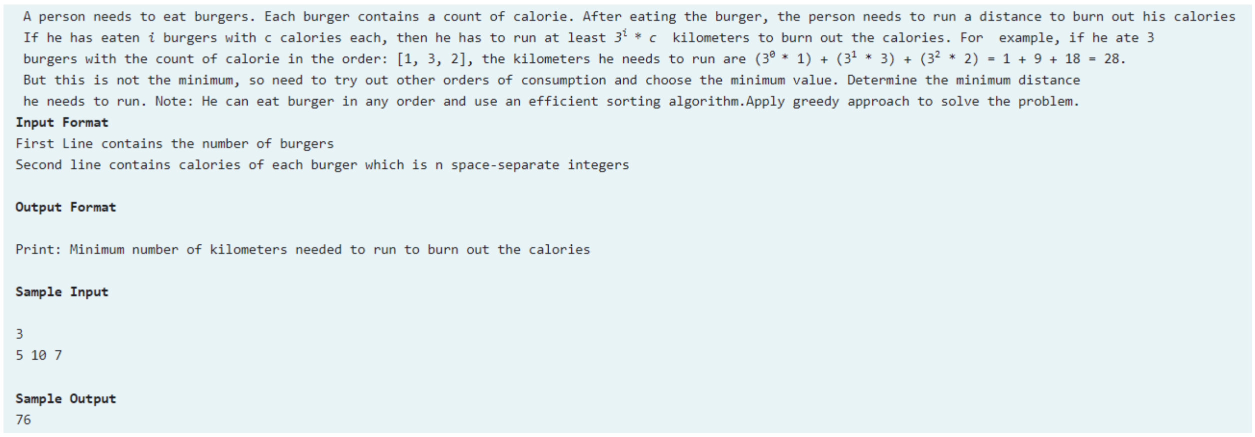
```
    for(int i=0;i<cookies;i++){
        scanf("%d",&ckgreed[i]);
    }
    int count=0;
    for(int i=0;i<child;i++){
        for(int j=0;j<cookies;j++){
            if(ckgreed[i]>=greed[i]){
                count++;
            }
            break;
        }
    }
    printf("%d",count);
}
```



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

Passed all tests! ✓

3.



A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person needs to run a distance to burn out his calories If he has eaten i burgers with c calories each, then he has to run at least $3^i * c$ kilometers to burn out the calories. For example, if he ate 3 burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 1 + 9 + 18 = 28$.
But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. Determine the minimum distance he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm.Apply greedy approach to solve the problem.

Input Format

First Line contains the number of burgers
Second line contains calories of each burger which is n space-separate integers

Output Format

Print: Minimum number of kilometers needed to run to burn out the calories

Sample Input

```
3
5 10 7
```

Sample Output

```
76
```

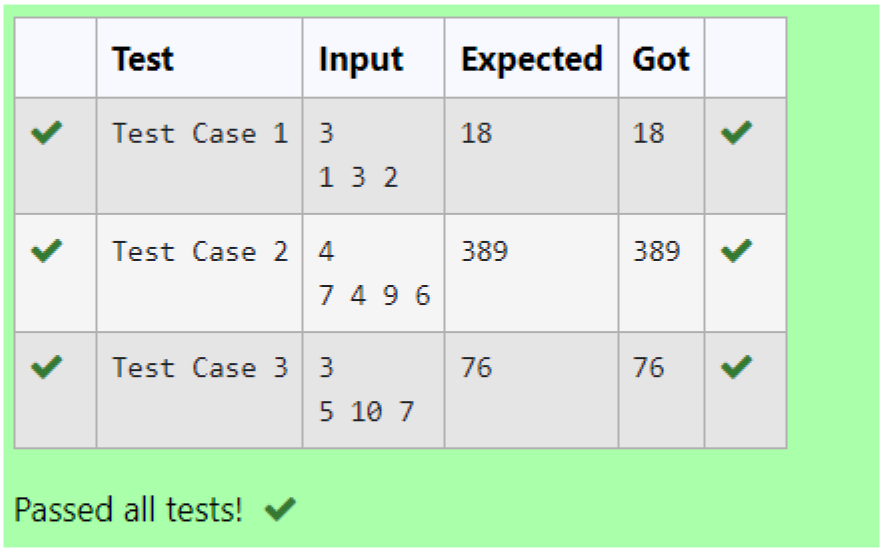
Algorithm:

1. Start
2. Input integer bgrs
3. Input array cal[bgrs]
4. Call bubbleSort(cal, bgrs)
5. Initialize km = 0, i = 0, and c = bgrs - 1
6. While c >= 0:
7. Update km += pow(bgrs, i) * cal[c]
8. Increment i, decrement c
9. Output value of km

Program:

```
#include<stdio.h>
#include <math.h>
void bubbleSort(int arr[], int n);
int main(){
  int bgrs;
  scanf("%d",&bgrs);
  int cal[bgrs];
  for(int i=0;i<bgrs;i++){
    scanf("%d",&cal[i]);
  }
  bubbleSort(cal,bgrs);
  int km=0;
  int i=0;
  int c=bgrs-1;
  while(c>=0){
    km+=pow(bgrs,i)*cal[c];
```

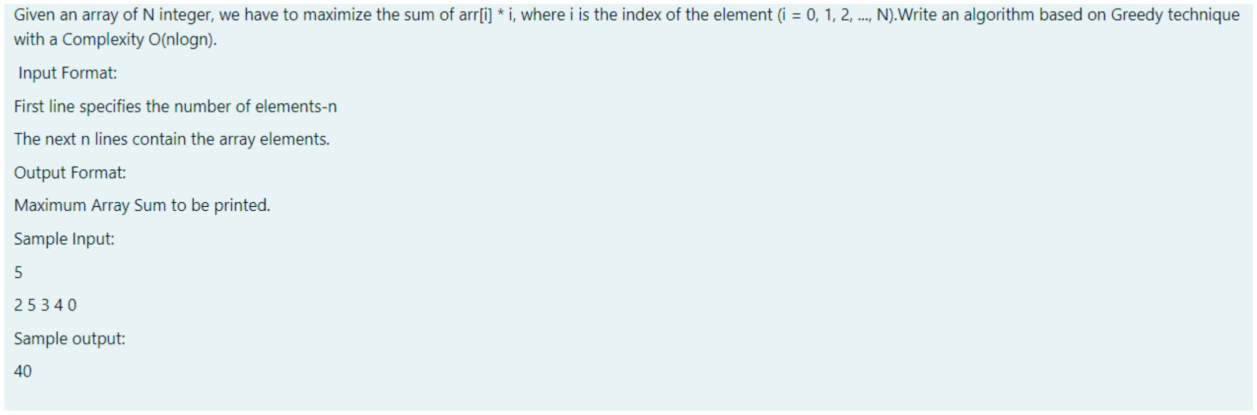
```
        i++;
        c--;
    }
    printf("%d",km);
}
void bubbleSort(int arr[], int n) {
    int i, j, temp;
    int swapped;
    for (i = 0; i < n-1; i++) {
        swapped = 0;
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                swapped = 1;
            }
        }
        if (swapped == 0) {
            break;
        }
    }
}
```



	Test	Input	Expected	Got	
✓	Test Case 1	3 1 3 2	18	18	✓
✓	Test Case 2	4 7 4 9 6	389	389	✓
✓	Test Case 3	3 5 10 7	76	76	✓

Passed all tests! ✓

4.



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

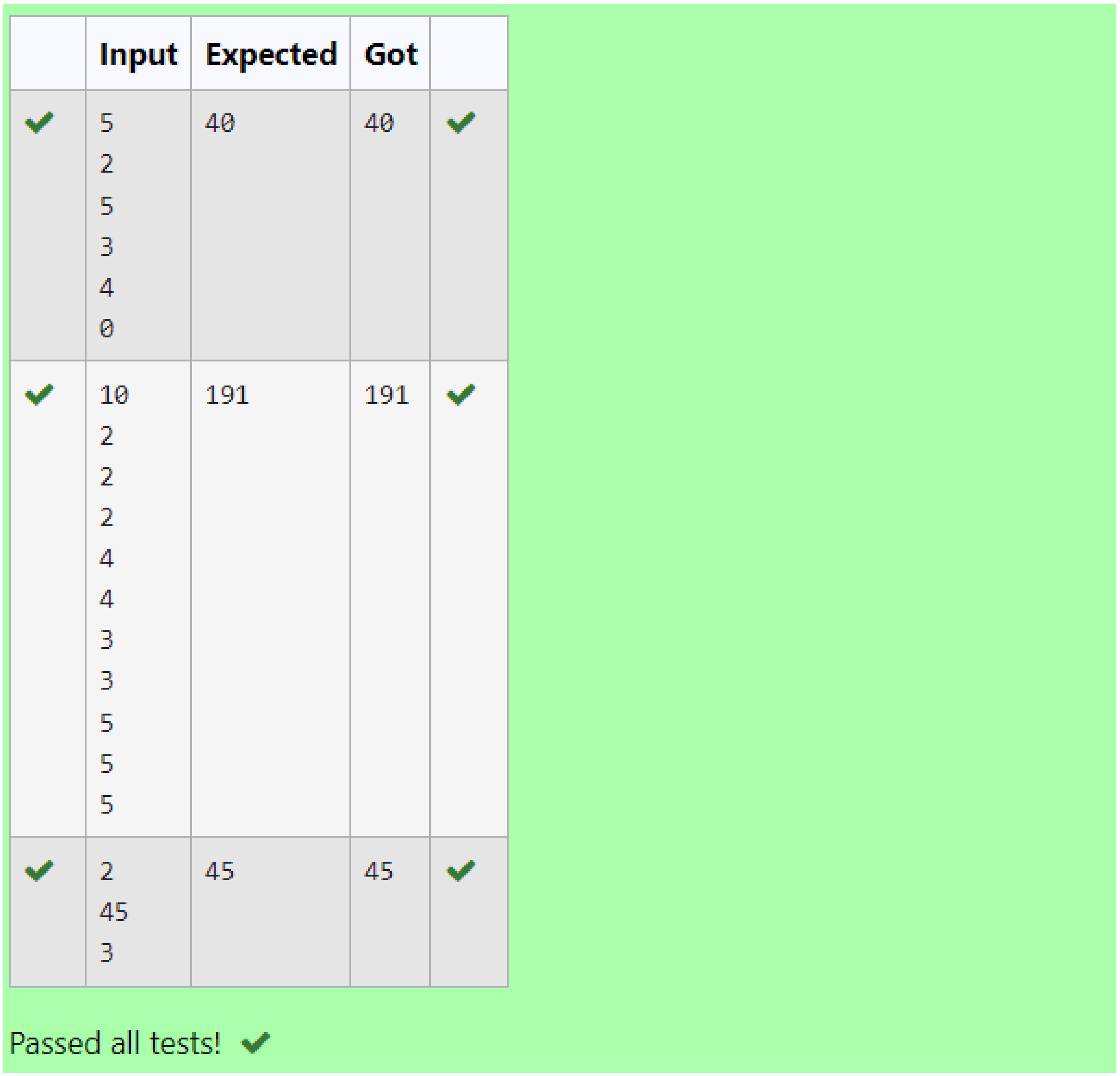
Algorithm:

```
1. Start
2. Input integer n
3. Input array a[n]
4. For i = 0 to n-1:
5. Input a[i]
6. For i = 0 to n-1:
7. For j = i+1 to n:
       a. If a[i] > a[j], swap a[i] and a[j]
8. Initialize sum = 0
9. For i = 0 to n-1:
10. Add a[i] * i to sum
11. Output value of sum
12. Sum
```

Program:

```
#include<stdio.h>
int main()
{
   int n;
   scanf("%d",&n);
   int a[n];
   for(int i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   int temp = 0;
```

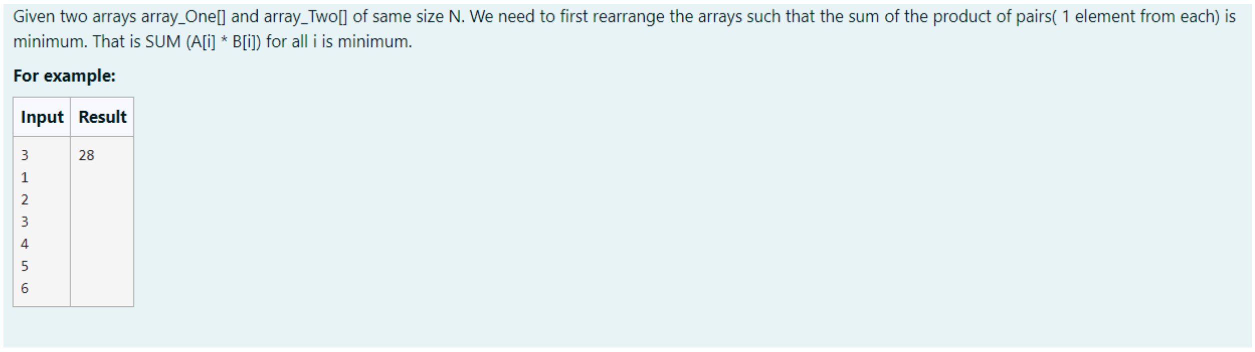
```
    for (int i = 0; i < n; i++)
    {
        for (int j = i+1; j < n; j++)
        {
            if(a[i] > a[j]) {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
    int sum=0;
    for(int i=0;i<n;i++)
    {
        sum+=(a[i]*i);
    }
    printf("%d",sum);
}
```



	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.



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

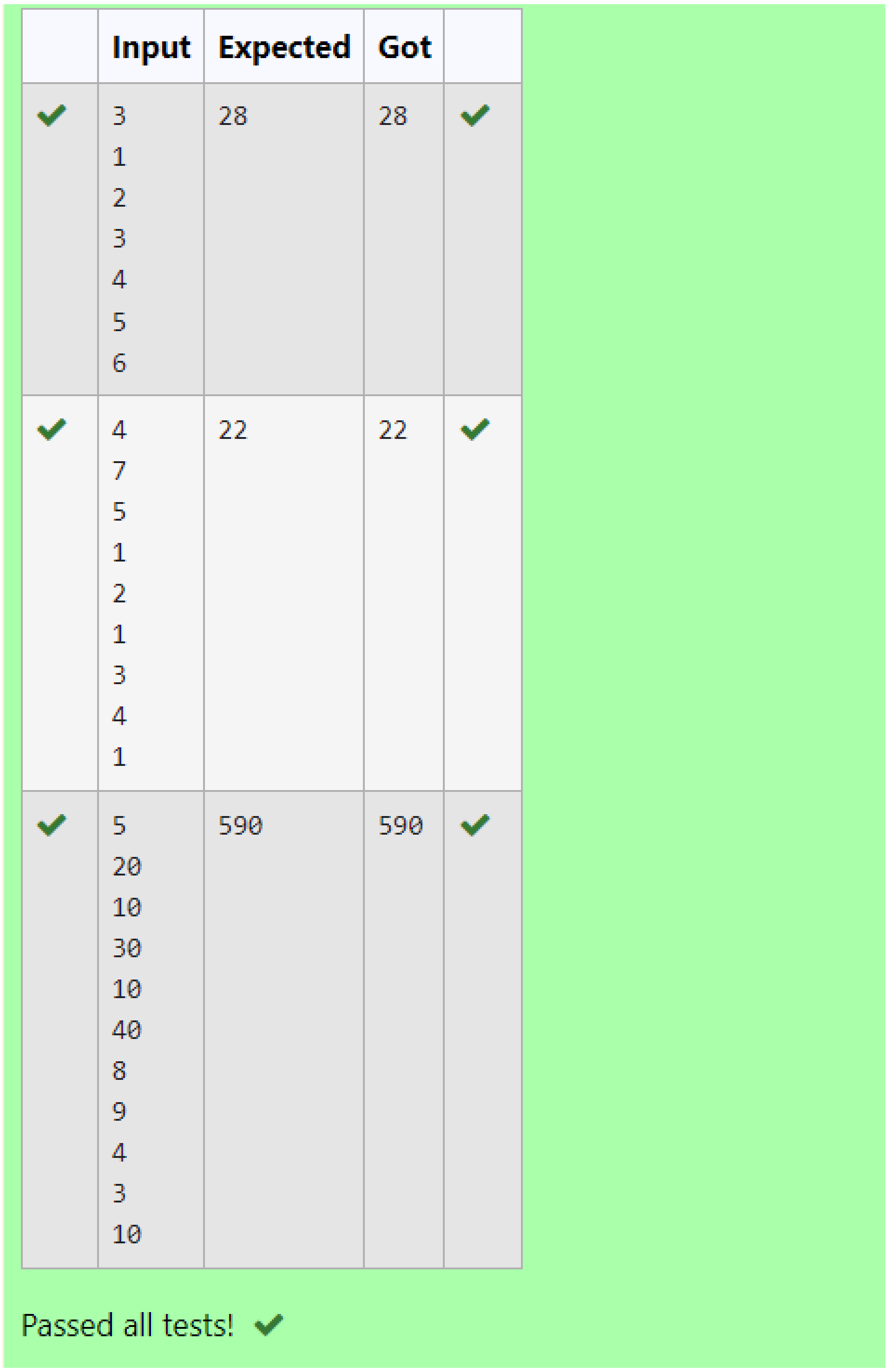
Algorithm:

13. Start
14. Input integer n
15. Input array a[n] and b[n]
16. Sort array a[] in ascending order
17. Sort array b[] in descending order
18. Initialize min = 0
19. For i = 0 to n-1:
20. Update min += a[i] * b[i]
21. Output value of min
22. End

Program:

```
#include<stdio.h>

int main()
{
   int n;
   scanf("%d",&n);
   int a[n],b[n];
   for(int i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   for(int i=0;i<n;i++)
   {
      scanf("%d",&b[i]);
   }
   int temp = 0;
```

```
    for (int i=0;i<n;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
    for (int i= 0; i < n; i++)
    {
        for (int j=i+1;j<n;j++)
        {
          if(b[i]<b[j])
          {
              temp=b[i];
              b[i]=b[j];
              b[j]=temp;
          }
        }
    }
    int min=0;
    for(int i=0;i<n;i++)
    {
        min+=(a[i]*b[i]);
    }
    printf("%d",min);
}
```



	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! ✓