```
Coders here is a simple task for you, you have given an array of size N and an integer M.
 Your task is to calculate the difference between maximum sum and minimum sum of N-M elements of the given array
 Constraints:
 1<=t<=10
 1<=n<=1000
1<=a[i]<=1000
  Input
 First line contains an integer T denoting the number of testcases.
First line of every testcase contains two integer N and M.
Next line contains N space separated integers denoting the elements of array
 Output:
 For every test case print your answer in new line
 SAMPLE INPUT
1
5 1
1 2 3 4 5
 SAMPLE OUTPUT
 Explanation
M is 1 and N is 5 to you have to calculate maximum and minimum sum using (5-1 =) 4 elements.

Maximum sum using the 4 elements would be (2-2+44-5-)14.

Minimum sum using the 4 elements would be (1-2-3+4-10).

Difference will be 14-10-4.
```



A new deadly virus has infected large population of a planet. A brilliant scientist has discovered a new strain of virus which can cure this disease. Vaccine produced from this virus has various strength depending on midichlorians count. A person is cured only if midichlorians count in vaccine batch is more than midichlorians count of person. A doctor receives a new set of report which contains midichlorians count of each infected patients are equal. First line contains the number of vaccines - N. Second line contains N integers, which are strength of vaccines. Third line contains N integers, which are midichlorians count of patients. Print a single line containing 'Yes' or 'No'. Input Constraint Strength of vaccines and midichlorians count of patients fit in integer. 123 146 454 542 456 SAMPLE OUTPUT

```
Answer (penalty repline 0 %)

2 tota main()

2 tota main()

3 tota main()

4 tota main()

5 tota main()

6 tota main()

7 for(tot log) tool)

8 score("m", social)

8 score("m", social)

10 score("m", social)

11 for(tot log) tool

12 total policy log

13 total policy log

14 for(tot log) tool

15 (

16 (social)

17 (social)

18 (ff(patk))

19 minch()

20 pervocalent

21 vec([]-tong)

22 topopolicy([]-tong)

23 vec([]-tong)

24 topopolicy([]-tong)

25 pervocalent

26 pervocalent

27 pervocalent

28 pervocalent

29 pervocalent

20 pervocalent

21 vec([]-tong)

22 topopolicy([]-tong)

23 pervocalent

24 pervocalent

25 pervocalent

26 pervocalent

27 pervocalent

28 pervocalent

29 pervocalent

20 pervocalent

20 pervocalent

21 pervocalent

22 pervocalent

23 pervocalent

24 pervocalent

25 pervocalent

26 pervocalent

27 pervocalent

28 pervocalent

29 pervocalent

20 pervocalent

21 pervocalent

22 pervocalent

23 pervocalent

24 pervocalent

25 pervocalent

26 pervocalent

27 pervocalent

28 pervocalent

29 pervocalent

20 pervocalent

21 pervocalent

22 pervocalent

23 pervocalent

24 pervocalent

25 pervocalent

26 pervocalent

27 pervocalent

27 pervocalent

28 pervocalent

29 pervocalent

20 pervocalent

21 pervocalent

22 pervocalent

23 pervocalent

24 pervocalent

25 pervocalent

26 pervocalent

27 pervocalent

27 pervocalent

28 pervocalent

29 pervocalent

20 pervocalent

20 pervocalent

20 pervocalent

2
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 int n,mind,min2,temp,flag=1;
scamf("%0",%n);
int vac[n],pat[n];
for(int i=0; i.no;i++)
scamf("%0",%vac[i]);
for(int i=0;i:n:i++)
scamf("%0",%pat[i]);
for(int i=0; j:n:1 ; j++)
{
min1-j,min2-j;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                       for (int job ; jon-1; jos) {
    intel_jmin2j;
    intel_jmin2j;
    intel_imin2j;
    intel_imin2j;
```

```
pat[min2]=pat[j];
pat[j]=temp;
 pat[j]=temp;
}
for (int i=0;i<n;i++){
    if(vac[i]<= pat[i]){
        flag=0;
        break;
    }
}</pre>
}
if(flag==1)
printf("Yes");
else
printf("No");
```

	Input	Expected	Got	
	Imput	Expected	-	
	5	No	No	~
	123 146 454 542 456			
	100 328 248 689 200			
ise	ed all tests! 🗸			

You are given an array of n integer numbers  $a_1, a_2, \ldots, a_n$ . Calculate the number of pair of indices (i, j) such that  $1 \le i < j \le n$  and  $a_i \times a_j = 0$ .

# Input format

- First line: n denoting the number of array elements - Second line: n space separated integers  $\sigma_1, \sigma_2, \ldots, \sigma_n$ .

## Output format

Output the required number of pairs.

1 ≤ a<sub>i</sub> ≤ 10<sup>9</sup>

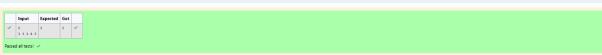
## SAMPLE INPUT

13143

### SAMPLE OUTPUT

### Explanation

The 2 pair of indices are (1, 3) and (2,5).



```
To are given an array A of non-negative integers of tipe at Your task it to ten't the array in non-decreasing order and print out the original indices of the new torsed array.

An ($5.2.7.1)

After sorting the new array becomes Av(1.3.6.5.7).

The required output should be "4.2.0.1.2."

INPUT:

The first line of input consists of the size of the array of size in

OUTPUT:

Coppet consists of a single line of integers

CONSTRAINTS:

1 - cm c - 164

6 - AAR( = 165

1 - MOTE. The indexing of the array starts with 0.

SAMPLE OUTPUT

4.2.0.1.3
```