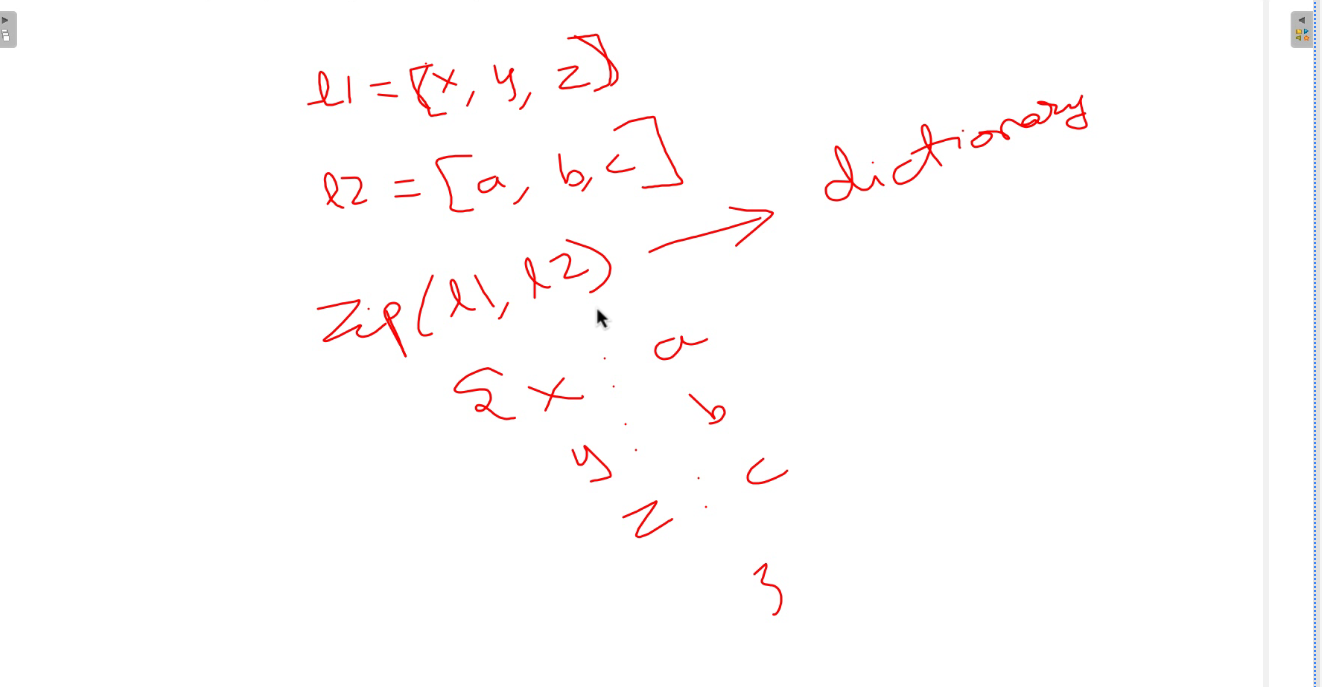
Pretty insights:



'''

You are provided with an array of integers called 'score,' with 'n' elements,

where 'score[i]' represents the performance score of the ith participant

in a competitive event. It is guaranteed that all the scores are distinct.

The participants' final positions are determined solely by their performance

scores, with the top performer ( with highest score), securing the 1st position,

the second-best performer obtaining the 2nd position, and so forth.

Each participant's rank is directly related to their placement:

- The participant in the 1st position is awarded the 'Champion'

- The participant in the 2nd position receives the 'RunnerUp-1'

- The participant in the 3rd position is granted the 'RunnerUp-2'

- For participants ranked from 4th to 'n,' their rank corresponds to their

position (i.e., the participant in the xth position is assigned the rank 'x').

Your task is to generate an array named 'answer' of size 'n'

Each element 'answer[i]' in this array should represent the rank achieved

by the ith participant.

Input Format:

-------------

Line-1: An integer N, number of participants.

Line-2: N space seprated integers, scores[].

Output Format:

--------------

Print the list of ranks of the participants.

Sample Input-1:

---------------

5

10 3 9 8 4

Sample Output-1:

----------------

[Champion, 5, RunnerUp-1, RunnerUp-2, 4]

Sample Input-2:

---------------

8

10 3 9 4 2 7 6 1

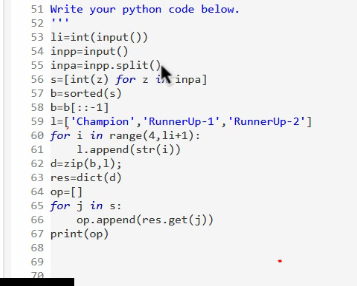
Sample Output-2:

----------------

[Champion, 6, RunnerUp-1, 5, 7, RunnerUp-2, 4, 8]

Write your python code below.

'''



You are provided a chocolate array, represented by the integer array chocolates",

which represents the number of chocolates in different boxes. Every second,

you perform the following steps:

1. Choose the box with the most number of chocolates.

2. If there are multiple boxes with the most chocolates,

you may select any one of them.

3. Take all the chocolates in that box, except for a square number of

chocolates that you leave behind.

4. Repeat the above steps for 'p' seconds.

You have to calculate how many chocolates left after p seconds.

Input Format:

-------------

Line-1: Two space separated integers, number of boxes N and number of seconds P.

Line-2: N space seprated integers, chocolates[].

Output Format:

--------------

Print an integer result, the number of chocolates left.

Sample Input-1:

---------------

5 4

16 100 625 81 9

Sample Output-1:

----------------

49

Explanation:

-------------

- In the first second, the middle box is chosen and 25 chocoloates are left behind.

- Then the second box is chosen and 10 chocoloates are left behind.

- After that the fourth box is chosen and 9 chocoloates are left behind.

- Finally, the middle box is chosen again and 5 chocoloates are left behind.

The final remaining chocoloates are [16,10,5,9,9],

so the total number of chocoloates remaining is 49.

Sample Input-2:

---------------

5 8

1 1 1 1 1

Sample Output-2:

----------------

5

My soln:

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int p=sc.nextInt();

int[] arr=new int[n];

for(int i=0;i<n;i++){

arr[i]=sc.nextInt();

}

for(int i=0;i<p;i++){

int j=max(arr);

// System.out.println(j);

arr[j]=(int)Math.sqrt(arr[j]);

// System.out.println("j"+arr[j]);

}

int sum=0;

for(int i=0;i<n;i++){

sum+=arr[i];

}

System.out.println(sum);

}

public static int max(int[] arr){

int max=0;

int max1=0;

for(int i=0;i<arr.length;i++){

if(arr[i]>max){

max=arr[i];

max1=i;

}

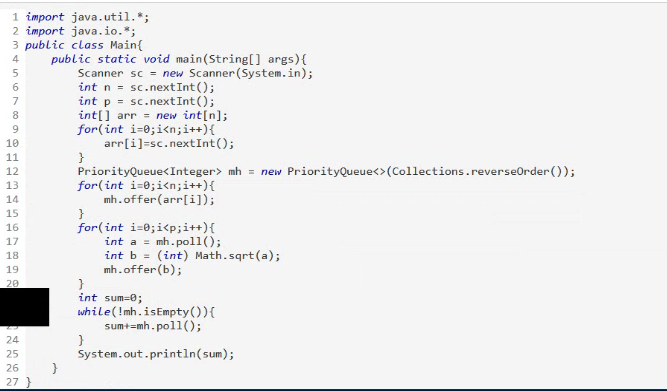
}

return max1;

}

}

Optimised way



Motu Patlu are good friends, Motu loves to eat Samosas,

He is given N Boxes of samosas[], box-'a' has samosas[a].

He can choose two boxes having highest number of samosas each time,

box-i and box-j, where samosas[i] <= samosas[j].

If samosas[i] == samosas[j] , then eat all the samosas from both boxes;

If samosas[i] != samosas[j] , then eat all samosas from box-i, and from

box-j eat only samosas[i] samosas, and left with ( samosas[j]-samosas[i] )

If the box becomes empty, remove the box.

At the end, there is at most 1 box left. Return the number of samosas left

in that box (or 0 if there are no boxes left.)

Input Format:

-------------

N space separated integers, number of samosas[i] in box[i]

Output Format:

--------------

Print number of the samosas left at the end.

Sample Input-1:

---------------

2 7 4 1 8 1

Sample Output-1:

----------------

1

Explanation:

------------

Boxes are numbered from 0,1,2,...,N-1.

Motu selects, box-1 has 7 samosas and box-4 has 8 samosas eat 14 samosas, boxes becomes [2,4,1,1,1]

Motu selects, box-0 has 2 samosas and box-1 has 4 samosas eat 4 samosas, boxes becomes [2,1,1,1]

Motu selects, box-1 has 1 samosa and box-0 has 2 samosas eat 2 samosas, boxes becomes [1,1,1]

Motu selects, box-0 has 1 samosa and box-1 has 1 samosa eat 2 samosas, boxes becomes [1]

Finally left with 1 box, box contains 1 samosa in it.

My soln(Optimized):

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

// ArrayList<Integer> k=new ArrayList<>();

String ss=sc.nextLine();

String[] s=ss.split(" ");

PriorityQueue<Integer> q=new PriorityQueue<>(Collections.reverseOrder());

for(int i=0;i<s.length;i++){

q.add(Integer.parseInt(s[i]));

}

while(q.size()>1){

int max1=q.remove();

int max2=q.remove();

if(max1==max2){

continue;

}

else{

q.add(Math.abs(max1-max2));

}

}

if(q.size()==1){

System.out.println(q.remove());

}

else{

System.out.println(0);

}

}

}