'''

A crew of N players played a game for one time, and got some scores.

They have to stand in the same positions after they played the game.

There is a constraint that, the player-K score, should not be greater or smaller than both of his neighbors.

To achieve this constraint, there are few steps to be followed as mentioned:

- If the score of player-K is smaller than both his neighbors,then his score is incremented by 1.

- If the score of player-K is greater than both his neighbors,then his score is decremented by 1.

- The first and last elements never change.

Repeat these steps, until the constraint is satisified.

Your task is to find and print the resultant array of scores, after the constraint is achieved.

NOTE: Players are not allowed to swap their positions to achieve the constraint.

Input Format:

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

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

Line-2: N space separated integers represemts scores of each player.

Output Format:

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

Print integer array, the resultant scores.

Sample Input-1:

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

6

4 3 5 2 6 3

Sample Output-1:

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

[4, 4, 4, 4, 4, 3]

Explanation:

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

Initially, the scores are changed from [4, 3, 5, 2, 6, 3] to [4, 4, 4, 3, 5, 3].

Next step, the scores are changed from [4, 4, 4, 3, 5, 3] to [4, 4, 4, 4, 4, 3].

No more operations can be done to this array.

Sample Input-2:

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

8

5 1 4 2 7 4 6 3

Sample Output-2:

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

[5, 3, 3, 3, 5, 5, 5, 3]

Sample Input-3:

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

10

48 71 26 73 53 98 1 68 50 18

Sample Output-3:

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

[48, 48, 49, 63, 63, 63, 50, 50, 50, 18]

Write your python code below

'''

n=int(input())

l=list(map(int,input().split()))

l2=list(l)

def fun(l,k):

while(k==True):

count=0

for i in range(1,len(l)-1):

if(l[i]<l[i-1] and l[i]<l[i+1]):

l2[i]+=1

# count+=1

elif(l[i]>l[i-1] and l[i]>l[i+1]):

l2[i]-=1

# count+=1

else:

count+=1

# l=l2

l=list(l2)

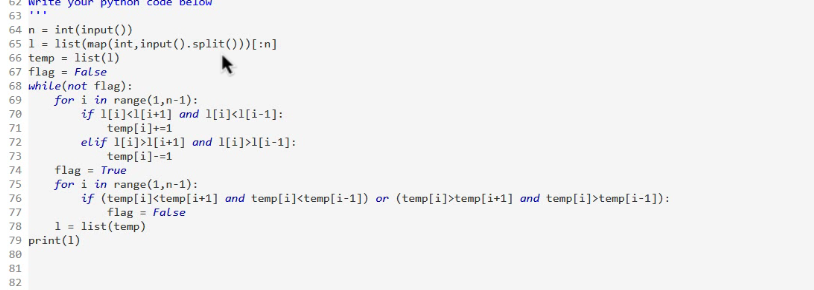
if(count==len(l)-2):

k=False

return l

l=fun(l,True)

print(l)



There are N sticks of various lengths, stklen[], where stklen[i] is the length

of the i-th stick. And you need to break them into pieces of any length. And

you will be given another integer P, number of pieces you need to make.

You need to break the sticks to make P pieces of equal lengths.

You can consider the unwanted piece as scrap.

You can break a stick of length 6 as follows:

- 6 pieces of length-1.

- 1 piece of length-1 and 1 piece of length-2 and one piece of length-3.

- 3 pieces of length-2, etc.

Your task is to check, can you break the sticks into P pieces of equal length,

if possible, retrun the maximum length of the pieces possible.

otherwise return 0.

Input Format:

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

Line-1: Two space separated integers, N and P.

Line-2: N space separated integers, length of the sticks.

Output Format:

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

Print an integer result.

Sample Input-1:

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

3 3

6 8 10

Sample Output-1:

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

6

Explanation:

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

- Keep the stick-1 as it is.

- Break the stick-2 into two pieces, one of length 6 and one of length 2.

- Break the stick-3 into two pieces, one of length 6 and one of length 4.

Finally, you have 3 pieces of length 6.

Sample Input-2:

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

3 4

6 8 10

Sample Output-2:

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

5

Explanation:

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

- Break the stick-1 into two pieces, one of length 5 and one of length 1.

- Break the stick-2 into two pieces, one of length 5 and one of length 3.

- Break the stick-3 into two pieces, one of length 5 and one of length 5.

Finally, you have 3 pieces of length 4.

Sample Input-3:

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

3 25

6 8 10

Sample Output-3:

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

0

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n1=sc.nextInt();

int n2=sc.nextInt();

int[] arr=new int[n1];

for(int i=0;i<n1;i++){

arr[i]=sc.nextInt();

}

int u=min(arr);

int l=0;

System.out.println(bs(arr,l,u,n2));

}

public static int bs(int[] arr,int l,int u,int n2){

int ans=0;

while(l<=u){

int mid=l+(u-l)/2;

// System.out.println(mid +" "+check(arr,mid));

if(check(arr,mid)>=n2){

ans=mid;

l=mid+1;

}

else {

u=mid-1;

}

// else{

// if(mid>ans){

// ans=mid;

// }

// }

}

return ans;

}

public static int check(int[] arr,int r){

int sum=0;

if(r!=0){

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

sum+=arr[i]/r;

}

}

return sum;

}

public static int min(int[] arr){

int min=Integer.MIN\_VALUE;

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

if(arr[i]>min){

min=arr[i];

}

}

return min;

}

}





Mr PushpaRaj has N bottles of Spirit as bottles[], where i-th bottle

contains bottle[i] liters of Spirit. He wants to make that all the bottles

contains same amount of Spirit. Generally, Spirit evaporates quickly,

due to this, P percentage of Spirit will be evaporated while pouring it

from one bottle to another, i.e., if He poured M liters of Spirit from

one bottle to another, the amount of Spirit evaporated is P% of M.

Your task is to help Mr PushpaRaj find the maximum amount of Spirit in each

bottle after making sure that all the bottles have the same amount of Spirit.

NOTE: Print the result within 10^(-5) of the actual result will be accepted.

HINT:

-----

Perform the binary search till the condition breaks,

high-low > 0.00001

Input Format:

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

Line-1: Two space separated integers, N and P.

Line-2: N space separated integers, bottles[].

Output Format:

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

Print a double value rounded to 5 decimals.

Sample Input-1:

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

4 75

2 8 3 12

Sample Output-1:

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

4.00000

Explanation:

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

::-The bottles contains {2,8,3,12},Pour 4 ltrs from bottles[1] to bottles[0].

4 \* 75% = 3 ltrs are evaporated and bottles[0] only got 4 - 3 = 1 ltr of Spirit.

::-The bottles contains {3,4,3,12}, Pour 4 ltrs from bottles[3] to bottles[0].

4 \* 75% = 3 ltrs are evaporated and bottles[0] only got 4 - 3 = 1 ltr of Spirit.

::-The bottles contains {4,4,3,8}, Pour 4 ltrs from bottles[3] to bottles[2].

4 \* 75% = 3 ltrs are evaporated and bottles[0] only got 4 - 3 = 1 ltr of Spirit.

::-All the bottles have 4 ltrs of Spirit in them, so return 4.

Sample Input-2:

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

4 50

3 5 8 10

Sample Output-2:

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

5.66666