'''

The heights of the students of two classes are given as h1[] and h2[],

the student count of the classes are same. The task is to find

the sum of the product of heights of these two classes.

For example, if h1 = [1,2,3,4] and h2 = [5,2,3,1], the sum of products would be

1\*5 + 2\*2 + 3\*3 + 4\*1 = 22.

You are given two lists h1 and h2 of length n, return the minimum product sum

if you are allowed to rearrange the order of the elements in h1.

Sample Input-1:

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

4

5 3 4 2

4 2 2 5

Sampe Output-1:

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

40

Explanation:

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

We can rearrange h1 to become [3,5,4,2].

The sum of products of [3,5,4,2] and [4,2,2,5] is 3\*4 + 5\*2 + 4\*2 + 2\*5 = 40.

Sample Input-2:

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

5

2 1 4 5 7

3 2 4 8 6

Sampe Output-2:

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

65

Explanation:

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

We can rearrange h1 to become [5,7,4,1,2].

The product sum of [5,7,4,1,2] and [3,2,4,8,6] is

5\*3 + 7\*2 + 4\*4 + 1\*8 + 2\*6 = 65.

'''

n=int(input())

h1=list(map(int,input().split()))

h2=list(map(int,input().split()))

l1=h1.copy()

l2=h2.copy()

d=dict()

sum=0

for i in range(len(l1)):

sum+=max(l2)\*min(l1)

# d[max(l2)]=min(l1)

l1.remove(min(l1))

l2.remove(max(l2))

# print(d)

print(sum)

Subodh is interested in playing with Strings,

For a given String 'S', Subodh applies some rules to find the value of 'S'.

The rules are as follows:

- If it is a balanced [] has value 1

- XY has value X + Y , where X and Y are balanced [] strings.

- [Z] has score 2 \* Z , where Z is a balanced parentheses string.

Find out the value of given String and print it.

Note: All the given strings are balanced

Input Format:

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

A String contains only '[', ']'

Output Format:

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

Print an integer as result.

Sample Input-1:

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

[]

Sample Output-1:

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

1

Sample Input-2:

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

[[]]

Sample Output-2:

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

2

Sample Input-3:

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

[[][[]]]

Sample Output-3:

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

6

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s=sc.next();

Stack<Integer> st=new Stack<>();

StringBuilder sb=new StringBuilder();

// int i=0;

int c=0;

int count=0;

st.push(0);

int ans=0;

int level=0;

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

if(s.charAt(i)=='['){

if(s.charAt(i+1)==']'){

ans+=Math.pow(2,level);

i+=1;

}

else{

level+=1;

}

}

else{

level-=1;

}

}

System.out.println(ans);

}

}

For X-Mas, santa claus is preparing a X-Mas Tree with set of Bulbs.

The bulbs are of different voltages, and preparation of tree as follows:

- The bulbs are arranged in level-wise, levels are numbered from 0,1,2,3..

so on.

- At level-0: There will be only one bulb as root bulb.,

- From next level onwards, we can attach atmost two bulbs to left side,

and right side of every bulb in previous level.

- The empty attachements in each level are indicated with null.

(for example: look in hint)

You will be given the root of the X-Mas Tree,

Your task is to findout the bulb with highest voltage in each level.

Implement the class Solution:

1.public List<Integer> maxInEachRow(BinaryTreeNode root): returns the list of integers.

Input Format:

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

A single line of space separated integers, voltages of the set of bulbs.

Output Format:

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

Print the list of voltages.

Sample Input-1:

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

2 4 3 6 4 -1 9

Sample Output-1:

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

[2, 4, 9]

Sample Input-2:

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

3 4 7 7 3 8 4

Sample Output-2:

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

[3, 7, 8]

/\*

//TreeNode Structure for Your Reference..

class Node{

public int data;

public Node left, right;

public Node(int data){

this.data = data;

left = null;

right = null;

}

}

\*/

import java.util.\*;

class Solution {

public List<Integer> maxInEachRow(Node root) {

List<Integer> l=new ArrayList<>();

Queue<Node> q=new LinkedList<>();

q.offer(root);

while(!q.isEmpty()){

int n=q.size();

int max=Integer.MIN\_VALUE;

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

Node x=q.poll();

if(x.left!=null){

q.offer(x.left);

}

if(x.right!=null){

q.offer(x.right);

}

max=Math.max(x.data,max);

}

if(max!=-1)

l.add(max);

}

return l;

}

}

You are given a set of N integers, and a value to find F. Initially a variable,

'total' is set to 0. You can perform either addition(+) or subtraction(-)

of every integer from the set to the 'total'. The resultant total should be

equal to the value F.

Your task is to find out the number of ways, you can make the 'total' equal to

the value F.

Input Format:

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

Line-1: Two integers N and F.

Line-2: N space separated integers

Output Format:

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

Print an integer, number of ways.

Sample Input:

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

5 3

1 1 1 1 1

Sample Output:

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

5

Explanation:

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

total = -1+1+1+1+1 = 3 -> total=value-F

total = +1-1+1+1+1 = 3 -> total=value-F

total = +1+1-1+1+1 = 3 -> total=value-F

total = +1+1+1-1+1 = 3 -> total=value-F

total = +1+1+1+1-1 = 3 -> total=value-F

NOTE: + means addition, - means subtraction

import java.util.\*;

public class Main{

// static int count=0;

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int f=sc.nextInt();

int[] arr=new int[n];

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

arr[i]=sc.nextInt();

}

int total=0;

int[] count=new int[1];

fun(arr,0,count,total,f);

System.out.println(count[0]);

}

public static void fun(int[] arr,int i,int[] count,int total,int f){

if(i==arr.length){

if(total==f){

count[0]+=1;

return;

}

return;

}

total+=arr[i];

fun(arr,i+1,count,total,f);

total-=arr[i];

total-=arr[i];

fun(arr,i+1,count,total,f);

total+=arr[i];

}

}