Problem C

Russian nesting dolls are brightly painted hollow wooden figures. The dolls in a set have roughly the same shape, typically humanoid, but different sizes. When the set is assembled, the biggest doll contains the second-biggest doll, the second-biggest contains the third-biggest, and so on.

We can approximate the shape of a doll as a cylinder of height h

, diameter d

, and wall thickness w

. Such a doll would have a hollow of height h−2w

and diameter d−2w

.

Boris and Natasha each has a set of dolls. The sets are nearly identical; each has the same number of dolls, which look the same but differ in their dimensions. Last night Boris and Natasha were playing with their dolls and left them in the living room. Their mother tidied them away, dumping them all in one box. Can you help Boris and Natasha separate their sets of dolls?

Input

Input will consist of several test cases (at most 5

). The first line of each test case will contain n

, the number of dolls in each set (1≤n≤200

). Then follow 2n

lines; each gives the integer dimensions, h

, d

, w

of a doll (0<2w≤h,d≤106

). A line containing 0

follows the last test case.

Output

For each test case, separate the dolls into two sets of nesting dolls such that, within each set, the dolls fit within each other, standing straight up, as described above. The first n

lines of output should give the dimensions of the dolls in one set, in decreasing order by height. The next line should contain a single hyphen, “-”. The next n

lines should give the dimensions of the dolls in the second set, also in decreasing order by height.

There will always be a solution. If there are many solutions, any will do. Output an empty line between test cases.

Sample Input 1Sample Output 13

100 100 3

97 97 3

94 94 3

91 91 3

88 88 3

85 85 3

5

100 100 1

97 97 3

98 98 1

96 96 1

94 94 1

92 92 1

90 90 1

88 88 1

86 86 1

84 84 1

0

100 100 3

94 94 3

88 88 3

-

97 97 3

91 91 3

85 85 3

100 100 1

98 98 1

96 96 1

94 94 1

92 92 1

-

97 97 3

90 90 1

88 88 1

86 86 1

84 84 1

Problem B

Kattis is taking one of her kittens out on several hiking trips, and they need to pack their backpacks. They have a number of items (tents, cooking equipment, food, clothing, etc.), and need to split the weight between them as evenly as possible. In case the weight can’t be split evenly, Kattis will carry the extra weight. Can you you help them split the items between them for each trip?

Input

Input contains of up to 150

hiking trips. Each trip is given as a line in the input. The line starts with 1≤n≤20

, the number of items they need to split. Then follows the weight of each item. The weights are all in the range of [100,600]

grams. End of input is indicated by a line containing a single 0

.

Output

For each trip, output the weights of the two backpacks. Output the weight of the backpack Kattis will carry first.

Sample Input 1Sample Output 18 529 382 130 462 223 167 235 529

12 528 129 376 504 543 363 213 138 206 440 504 418

0

1344 1313

2181 2181

solution B

input\_sample = [8, 529, 382, 130, 462, 223, 167, 235, 529, 12, 528, 129, 376, 504, 543, 363, 213, 138, 206, 440, 504, 418, 0]def sumSplit(left,right=[],difference=0):

sumLeft,sumRight = sum(left),sum(right) # stop recursion if left is smaller than right

if sumLeft<sumRight or len(left)<len(right): return # return a solution if sums match the tolerance target

if sumLeft-sumRight == difference:

return sumLeft, sumRight # recurse, brutally attempting to move each item to the right

for i,value in enumerate(left):

solution = sumSplit(left[:i]+left[i+1:],right+[value], difference)

if solution: return solution if right or difference > 0: return

# allow for imperfect split (i.e. larger difference) ...

for targetDiff in range(1, sumLeft-min(left)+1):

solution = sumSplit(left, right, targetDiff)

if solution: return solutionweight\_list = []

count\_list = []

result = []for item in input\_sample:

if 1 <= item <= 20:

count\_list.append(item)

elif 100 <= item <= 600:

weight\_list.append(item)

elif item == 0:

breakresult = []for n in range(len(count\_list)):

calculate\_list = weight\_list[:count\_list[n]] result.append(sumSplit(calculate\_list)) del weight\_list[:count\_list[n]]result = [(1344, 1313), (2181, 2181)]

sum = []

for i in result:

sum.extend(i)for i in range(len(sum)):

if i % 2 == 0:

print(sum[i], sum[i+1])

Problem A

The ancient and mysterious Apaxian civilization, which we most certainly did not make up, continues to confound the researchers at the Oriental Institute. It turns out that the Apaxians had a peculiar naming system: the more letters in your name, the higher your status in society. So, in Apaxian society, bob was probably a lowly servant, and bobapalaxiamethostenes was likely a High Priest or Minister. Even more than that, Apaxians valued the number of adjacent letters that were the same in a name. So, while bob continues to be an unimpressive name, boooooooooob probably elicited cheers and applause wherever he went.

Unfortunately, this makes the task of reading Apaxian scrolls very cumbersome, specially when you consider that a particularly famous Apaxian queen had ten thousand consecutive a’s in her name. Legend has it that she was already two years old by the time the Royal Herald finished announcing her birth.

To make the Oriental Institute’s life easier, the Department of Computer Science has offered to convert the Apaxian scrolls into a more readable format. Specifically, we will be taking Apaxian names and replacing all consecutive runs of the same letter by a single instance of such letter.

So, for example, the compact version of boooob would be bob, where the four consecutive o’s have been replaced with a single o. Similarly, the compact version of bbbooobbb would also be bob. On the other hand, the compact version of bob is still bob.

Input

The input contains a single name. Each name contains only lowercase letters (a–z), no whitespace, a minimum length of 1 character, and a maximum length of 250 characters.

Output

The output contains the compact version of the name: any time the same letter appears two or more times in sequence, it must be replaced by a single instance of that letter.

Solution A

a = input()

n = ‘’

if len(a) == 1:

print(a)

else:

for i in range(len(a)-1):

if a[i] != a[i+1]:

n += a[i]

n += a[-1]

print(n)