Problem A. Inspection

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 mebibytes

Due to an inspection in an Imperial Army, it was decided to upgrade the droid army. Emperor Palpatine ordered to throw away two droids with the lowest serial number. Find them!

Input

First line contains an integer N — number of droids ($2 \le N \le 100\,000$), second line contains N integers, not exceeding $2 \cdot 10^9$ in absolute value — droids' serial numbers.

Output

Output two integers, first one should be the minimum serial number, second one — the second minimum serial number.

Examples

standard input	standard output
5 49 100 23 -100 157	-100 23
3 1 2419 1	1 1

Problem B. Sorting

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	64 mebibytes

Sort the given array of integers. **Don't use the standard sorting algorithm, code your own.**

Input

First line contains one integer N ($1 \le N \le 1000$), second line — N integers, not exceeding $2 \cdot 10^9$ in absolute value.

Output

Print given integers in non-decreasing order.

Examples

standard input	standard output
5 9 2 7 1 2	1 2 2 7 9

Problem C. Inversions

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 mebibytes

Write a program that for a given array $A = \langle a_1, a_2, \ldots, a_n \rangle$ finds the number of pairs of indices (i, j), such that $i < j$ and $a_i > a_j$.

Input

The first line of input contains a single integer n ($1 \le n \le 5\,100$), the number of elements in the array. The next line contains n pairwise distinct integers $a_1, \ldots, a_n$ ($0 \le a_i \le 10^9$).

Output

Print a single integer: the answer to the problem.

Example

standard input	standard output
5 4 3 2 5 1	7

Problem D. Selection sort

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Implement the selection sort, which, on each step, finds the minimum element in the remaining part of the array and puts it in the first position. Print the number of times the algorithm actually swapped two elements (don't count situations when the minimum is already in the first place).

Input

First line contains n ($1 \le n \le 1000$) — size of the array. Next line contains n distinct numbers from 1 to n, numbers are separated by spaces.

Output

Output one number — the number of pairs that were swapped during sorting.

Example

standard input	standard output
4 4 2 1 3	2

Problem E. Insertion sort

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Implement the insertion sort algorithm. Print the number of pairs of adjacent elements that were swapped during the sorting process.

Input

First line contains n ($1 \le n \le 1000$) — size of the array. Next line contains n distinct numbers from 1 to n, numbers are separated by spaces.

Output

Output one number — the number of pairs that were swapped during sorting.

Example

standard input	standard output
4 4 2 1 3	4

count

Problem F. Archive creation

sum < S

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	64 mebibytes

System administrator decided to make a backup of users' data. Unfortunately, hard drive size is less then sum of sizes of all archived files.

You know the sizes of all files. Write a program, that given information about users' data and hard drive size finds the maximum number of users, that can have their data backed up.

Input

First line contains integers S — size of the hard drive ($0 \le S \le 10^7$) and N — number of users ($1 \le N \le 10^5$). Then N positive integers not exceeding 300 follow, size of data for each user.

Output

Output the maximum number of users that can have their data backed up.

300'000

sort

Examples

standard input	standard output
100 2 200 50	1
100 3 50 30 50	2

S long long
N long long

cin << S << N;
for i < N:
cin << arr[i]
sort arr
count = 0;
sum = 0;
for i < N
if sum < S.
count ++.

100

25

3

Problem G. Intricate sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 mebibytes

You are given a sequence of numbers. Sort these numbers in increasing order of their last digit. If last digits of two numbers are the same, sort them by their value.

Input

First line contains N, the length of the sequence ($1 \le N \le 100$). Next N lines contains numbers in sequence, each on its own line. Numbers are positive and do not exceed 32000.

Output

Output a sequence of numbers, sorted according to the rule above.

Example

standard input	standard output
3 1280 382 930	930 1280 382 0 0 2

size ~~elem~~ : int

elem : long long

→ compare by mod 10, if equal compare by value.

9 % 10

9 < 6 < 6 %

100

75

25

< <=

75

+25

Problem H. Digital root

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	64 mebibytes

A *digital root* of number n can be determined as follows: take sum of digits of n, then take sum of digits of that number and so on, until you get a 1-digit number, which will be a digital root.

Your task is to sort the given array in increasing order of their digital root. If digital roots of two number are the same, the lesser number should go before the larger one.

Input

First line of input file contains elements of the array. Array length does not exceed 200, each number of positive and does not exceed 10^9.

Output

Output elements of the array, sorted by their digital root, in case of equality, by the value of the number.

Examples

standard input	standard output
15 14 13 12 11 10 9 8 7	10 11 12 13 14 15 7 8 9
80 61 51 41 22 1	1 22 41 51 61 80

size : int

elem : long long

compare by fn.

sum = 0.

fn — while n > 10.

~~sum~~

while n > 1

sum += (n % 10)

n = n/10

Problem I. Union of segments

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 mebibytes

Solving a problem from a math test, Vasya found the answer as a union of N segments $[L_i, R_i]$ on a number line. But, some of these segments can intersect, which Vasya doesn't quite like. Your task is to represent Vasya's answer as a union of a minimum number of non-intersecting segments.

Input

First line contains one integer N ($1 \le N \le 50000$). Next N lines contains pairs of integers L_i and R_i ($|L_i|, |R_i| \le 50000$), describing segments.

Output

In the first line, output a single integer M — the number of segments in the union. Next M lines should contain segments in the same format, as in the input file. Left endpoints of these segments must be in increasing order.

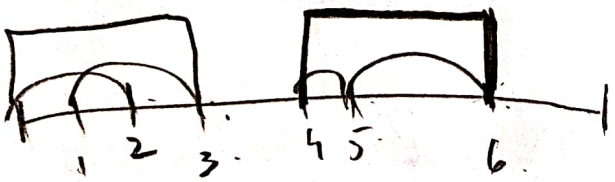
Example

standard input	standard output
4 0 2 4 5 1 3 5 6	2 0 3 4 6

sort based on left and right

~~covered = a[0][0]~~ covered = a[0][1]
ans Vector append (a[0])

for (i = begin ; i < end ; i++)
if a[i][0] >= covered
if (i != begin)
ans Vector append (a[i-1]
covered = a[i-1][1]



Problem J. Minimal cover

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	64 mebibytes

You are given some set of segments on a number line with integer endpoints $[L_i, R_i]$. Choose the minimum number of segments that cover $[0, M]$ completely (M is natural number).

Input

First line contains integer M ($1 \le M \le 5000$). Next lines describe segments. Each line contains a pair of integers L_i and R_i ($|L_i|, |R_i| \le 50000$), the left and right endpoint of a segment. Input ends with a pair of zeros (doesn't count as a segment). Total number of segments is less than or equal to 100 000.

Output

In the first line output the minimum number of segments, needed to cover $[0, M]$ completely. Next, output segments that cover everything in the same format, as in input file (except for two zeros in the end). Output segments in order of increasing left endpoint.

, . , e. .

If it's impossible to cover $[0, M]$ using given segments $[L_i, R_i]$ print "No solution".

Examples

standard input	standard output
1 -1 0 -5 -3 2 5 0 0	No solution
1 -1 0 0 1 0 0	1 0 1

first insert
may miss
something

(( ))

[ ] [ ]

0 1

5
-1 0
0 2
1 5
2 3
0 0

before print
check covering

?

if

Problem J. Minimal cover

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	64 mebibytes

You are given some set of segments on a number line with integer endpoints $[L_i, R_i]$. Choose the minimum number of segments that cover $[0, M]$ completely (M is natural number).

Input

First line contains integer M ($1 \le M \le 5000$). Next lines describe segments. Each line contains a pair of integers L_i and R_i ($|L_i|, |R_i| \le 50000$), the left and right endpoint of a segment. Input ends with a pair of zeros (doesn't count as a segment). Total number of segments is less than or equal to 100 000.

Output

In the first line output the minimum number of segments, needed to cover $[0, M]$ completely. Next, output segments that cover everything in the same format, as in input file (except for two zeros in the end). Output segments in order of increasing left endpoint.

, . , e. .

If it's impossible to cover $[0, M]$ using given segments $[L_i, R_i]$ print "No solution".

Examples

standard input	standard output
1 -1 0 -5 -3 2 5 0 0	No solution
1 -1 0 0 1 0 0	1 0 1

R

sort based on left

filter the ones ≤ current L

pick the highest $[l_L, r_L)$

current L = r L

if L > M end

current L = snd

if a[i] > max

max = a[i].snd

CurrentL = a.first

prev L = current L then break

filter > init L

init L = current L.

No solution