Problem A. Automatic Fare System

Input file:	`automatic.in`
Output file:	`automatic.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Transport authorities of Sin-Andrewsburg have decided to introduce zone-based payment for travelling by city metro. They have hired you to develop automatic ticket billing system.

Sin-Andrewsburg metro has n lines, the i-th line has c_i stations numbered from 1 to c_i along the line. Some stations are transfer stations where passengers can change from one line to another. There are t transfer stations. It is possible to get from any station to any other using lines and changing from one line to another at transfer stations.

After introducing fare zones stations of each line are now divided to several segments that belong to various zones. Zones are numbered from 1 to z. Line i is divided to k_i segments, the j-th segment includes stations from $l_{i,j}$ to $r_{i,j}$ that belong to zone $z_{i,j}$. Zones at all lines are organized in such way that adjacent segments have zone numbers that differ by 1. Zone number decreases as the line goes toward city center and increases then, so for each line sequence $z_{i,j}$ satisfies the following: $z_{i,1} > z_{i,2} > \ldots > z_{i,f_i} < z_{i,f_i+1} < \ldots < z_{i,k_i}$ for some f_i from 1 to k_i. Transfer stations belong to the same zone at all lines they allow to change among.

Automatic fare system must analyze data about the station where the metro trip starts and the station where the trip ends and calculate the fare. The fare is proportional to the minimal number of zones that must be visited during the trip. If some zone is entered and exited several times during the trip, it is still counted once.

You must create a program that given information about the metro map, fare zones and trip start and end stations would find the number of zones that must be visited during the trip.

Input

The first line of the input file contains two integers: n and z — the number of lines and the number of zones ($1 \le n \le 10$, $1 \le z \le 100\,000$). Descriptions of lines follow.

Each description starts with c_i — the number of stations on the line ($1 \le c_i \le 10^9$). Then the number of zone segments on the line k_i follows ($1 \le k_i$, the sum of all k_i doesn't exceed 100 000). The following k_i lines contain information about zone segments. Each line contains three integers: $z_{i,j}$, $l_{i,j}$ and $r_{i,j}$ ($1 = l_{i,1} \le r_{i,1}$; $r_{i,1} + 1 = l_{i,2} \le r_{i,2}$; $\ldots$; $r_{i,k_i-1} + 1 = l_{i,k_i} \le r_{i,k_i} = c_i$; $z_{i,1} > z_{i,2} > \ldots > z_{i,f_i} < z_{i,f_i+1} < \ldots < z_{i,k_i}$; $|z_{i,j} - z_{i,j+1}| = 1$).

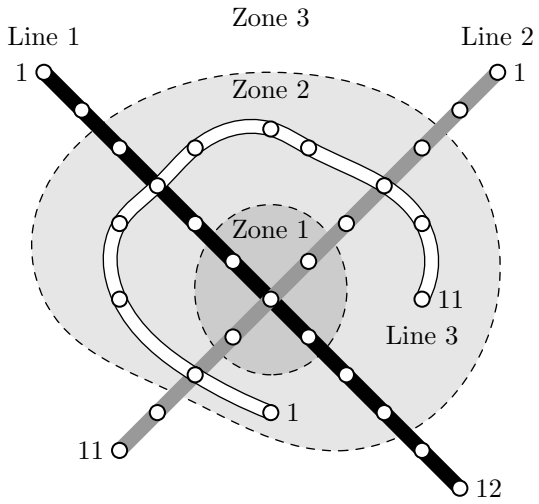
The following line contains t — number of transfer stations ($0 \le t \le 30$). Descriptions of transfer stations follow. Each description starts with v_i — number of lines that pass through the corresponding transfer station. It is followed by v_i lines, each line contains two integers x_i and s_i — number of the line and number of the station on the line that belongs to this transfer station. Each station on each line belongs to at most one transfer station. All stations that allow transfer to each other belong to the same zone.

The following line contains q — the number of trips to analyze, followed by the descriptions of the trips ($1 \le q \le 100\,000$). Each trip is described by four integers: a_i, sa_i, b_i, sb_i — the number of the line where the trip starts, the number of the station on this line, the number of the line where the trip ends, the number of the station on this line ($1 \le a_i, b_i \le n$, $1 \le sa_i \le c_{a_i}$, $1 \le sb_i \le c_{b_i}$).

Output

For each trip output the minimal number of fare zones that must be visited to get from the start station to the end station of the trip. Note, that you must visit at least one zone even if there is transfer between these stations and no train trip is necessary.

Examples

automatic.in	automatic.out
3 3 12 5 3 1 2 2 3 5 1 6 8 2 9 10 3 11 12 11 5 3 1 2 2 3 5 1 6 8 2 9 9 3 10 11 11 1 2 1 11 4 2 1 7 2 7 2 1 4 3 5 2 2 4 3 9 2 2 9 3 2 12 1 1 1 2 1 1 1 7 1 1 1 12 1 1 2 1 1 1 2 4 1 1 2 7 1 1 2 11 1 1 3 1 1 1 3 11 2 1 2 11 3 4 1 9 2 8 1 6	1 3 3 2 2 3 2 2 2 2 2 1 

Problem B. Bubble Sort

Input file:	bubble.in
Output file:	bubble.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Bubble sort is a well known sorting algorithm that can be described by the following pseudocode:

```
define BubbleSort(a: array [1..n]):
  for i from 1 to n − 1:
    for j from 1 to n − 1:
      if a[j] > a[j + 1]:
        swap(a[j], a[j + 1])
```

Let us call one iteration of the external cycle a *bubble pass*.

```
define BubblePass(a: array [1..n]):
  for j from 1 to n − 1:
    if a[j] > a[j + 1]:
      swap(a[j], a[j + 1])
```

Actually, not all of $n - 1$ bubble passes in the original algorithm are always necessary. Some arrays get sorted after fewer passes. For example, array $\langle 2, 1, 4, 3 \rangle$ is sorted by just one bubble pass.

Actually there are many arrays that can be sorted by just one bubble pass. For example, 4 out of 6 permutations of numbers from 1 to 3 are sorted by a single bubble pass. Let us denote the number of permutations of numbers from 1 to n that are sorted by a single bubble pass as $B(n)$.

Given n and k you have to find the k-th lexicographically permutation of numbers from 1 to n that is sorted by a single bubble pass.

Input

The input file contains several test cases.

Each test case consists of two integers n and k on a line ($1 \le n \le 100\,000$, $1 \le k \le 5 \cdot 10^{18}$, k doesn't exceed $B(n)$).

The last line of the input file contains two zeroes, it must not be processed.

The sum of n for all test cases in the input file doesn't exceed 10^6.

Output

For each test case in the input file output one line that contains n integers — the k-th lexicographically permutation of numbers from 1 to n that is sorted by a single bubble pass.

Examples

bubble.in	bubble.out
3 1 3 2 3 3 3 4 0 0	1 2 3 1 3 2 2 1 3 3 1 2

Problem C. Comb Avoiding Trees

Input file:	`comb.in`
Output file:	`comb.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

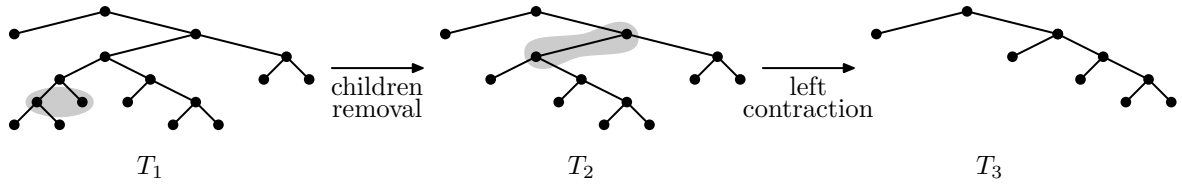
Structures that avoid some substructures are widely studied in combinatorics. In this problem we are interested in trees avoiding some special type of subtrees.

Consider rooted binary trees in which each node either has exactly two children: left and right (internal node), or has no children (leaf). In a special case of one-node tree its root is considered to be a leaf. A tree T is said to contain tree R as a *contraction subtree* if one can obtain R from T by a sequence of the following transformations:

- *Children removal*: remove both children of internal node obtaining a new leaf.
- *Left contraction*: let y be the left child of x. Replace children of x by children of y.
- *Right contraction*: let y be the right child of x. Replace children of x by children of y.

A tree T is said to *avoid* tree R if T doesn't contain R as a contraction subtree.

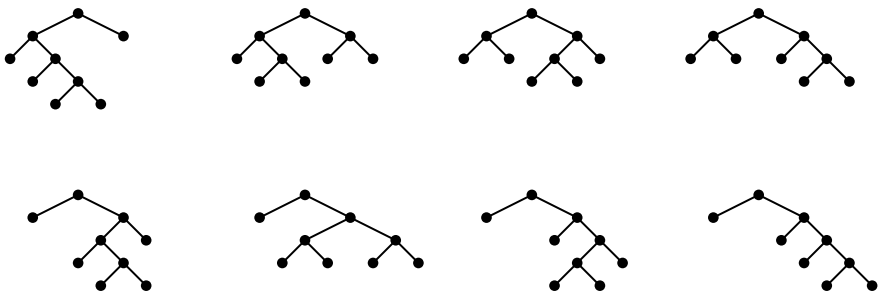
Examples of tree transformations are shown on the picture below. The picture also demonstrates, that tree T_1 contains tree T_3 as a contraction subtree.



Left k-comb is a tree with k leaves where right child of any node is a leaf. The following picture shows left k-combs for k from 2 to 5.

Given k and n you have to find for all i from 1 to n the number of trees that have i leaves and avoid left k-comb. Output all numbers modulo $10^9 + 9$.

All trees avoiding left 4-comb and containing 5 leaves are shown on the picture below.



Input

The input file contains two integers: k and n on a line ($2 \le k \le 5000$, $1 \le n \le 5000$).

Output

Output n integers: for each i from 1 to n print the number of trees that have i leaves and avoid left k-comb modulo $10^9 + 9$.

Examples

comb.in	comb.out
4 5	1 1 2 4 8
7 6	1 1 2 5 14 42

Problem D. Defend the Tower

Input file:	`defend.in`
Output file:	`defend.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Den is planning to take part in International Tower Defense Contest. The contest proceeds as follows.

The battlefield is a line of 10 cells numbered from 1 to 10, each cell can be guarded by some tower. There are 10 monsters with h hit points each. The game proceeds discretely by seconds. At the beginning of each second from 1 to 10 a new monster enters the first cell. Each monster has special parameter called *delay*, by default it is equal to 1. Monster's delay can be increased by *slow* towers. If monster's delay is d and it entered its current cell at the beginning of the t-th second, it moves to the next cell at the beginning of the $t + d$-th second.

There are three types of towers:

- *Gun tower* — every second it fires at monster on the cell guarded by it. If there are several monsters, only one of them is damaged, the one that entered the battlefield first. Gun tower causes damage of 1 hit point and costs 3 dollars. Gun tower is denoted by '`G`'.
- *Rocket tower* — fires every third second (so it fires on seconds 3, 6, 9, etc). It damages all monsters on the guarded cell. Rocket tower causes damage of 4 hit points. Rocket tower costs 10 dollars. Rocket tower is denoted by '`R`'.
- *Slow tower* — each slow tower among the last 5 cells visited by the monster increases its delay by 1. So, for example, if there are slow towers on cells 1, 2 and 4 monsters at cell 6 have delay equal to 3. Slow tower costs 5 dollars. Slow tower is denoted by '`S`'.

If the monster tries to move from cell 10, the monster is considered to pass and leaves the battlefield. If monster's hit points drop to zero or below, the monster is considered to be killed. The goal of the game is to set up towers in order to kill maximal number of monsters. Towers must be set up before the game starts.

Depending on the money the player has, he can kill different number of monsters. Help Den find out what maximal number of monsters he can kill depending on his money. For each sum of money s such that it is possible to kill strictly more monsters by setting up towers with total cost s than by towers with total cost $s - 1$, print information about this number of monsters and the optimal line of towers.

Input

The input file contains one integer h ($1 \le h \le 30$).

Output

For each sum of money s such that it is possible to kill k monsters by setting up towers with total cost s, and k is strictly greater than the number of monsters that can be killed by towers with total cost $s - 1$, print one line.

The line must have format "`$`s `-> kill` k `with` t" where t is the description of towers that must guide cells from 1 to 10. Use '`.`' to denote unguarded cell. If there are several rows of towers that allow to kill k monsters with total cost s, output lexicographically smallest description of such row.

Examples

defend.in	defend.out
13	$27 -> kill 1 with ...SSSGGGG $33 -> kill 4 with ..R..R..RG $38 -> kill 7 with ....GSSSRR $41 -> kill 8 with ..SSSRRGG. $43 -> kill 10 with ....GSSRRR

Problem E. Exam Scoring

Input file:	exam.in
Output file:	exam.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Andrew is teaching Algorithms and Data Structures course in St Petersburg National Search University for Information Nanotechnologies, Picomechanics and Femtooptics. Today he is running a written exam for his students. Exam consists of n tasks that must be solved and submitted for evaluation.

Andrew has checked all students' examination works and found out for each student and each task whether he/she has solved it. Now he would like to assign points to each task so that harder tasks solved by fewer students got more points. The points assigned to tasks must sum up to a total score of s.

Formally, let a_i be the number of students that solved the i-th task, let $a_i > 0$ for each i. Andrew would like to find such b_i that $a_i b_i = c$ for all i where c is constant and $\sum_{i=1}^{n} b_i = s$.

Unfortunately such b_i may be fractional. Andrew doesn't like fractional numbers, so he would like to approximate b_i by integer p_i using least squares method. Also Andrew thinks that scoring some task too low or too high is not fair. So points for each task must be between L and U, inclusive, for given L and U.

So now Andrew needs to find integer values p_i such that $L \le p_i \le U$ for all i, $\sum_{i=1}^{n} p_i = s$, and $\sum_{i=1}^{n} (b_i - p_i)^2$ is minimal possible.

Andrew is tired by checking students' works, so he asks you, his assistant, to do the job of assigning points to tasks.

Input

The input file contains several test cases.

The first line of each test case contains two integers: n and s ($1 \le n \le 100$, $n \le s \le 10^9$). The second line contains n integers: $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 18$). The third line contains two integers: L and U ($1 \le L \le U \le s$, $Ln \le s \le Un$).

The last test case is followed by a line containing two zeroes, it must not be processed. The sum of n in all test cases doesn't exceed 10 000.

Output

For each test case output n integers: p_i that must be assigned to tasks.

Examples

exam.in	exam.out
4 100 1 2 3 7 10 30 4 100 1 2 3 7 20 50 0 0	30 30 25 15 40 20 20 20

Problem F. Frequent Permutations

Input file:	`frequent.in`
Output file:	`frequent.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Peter is developing new casino software. Now he is writing a part of software that will shuffle the cards deck. To do so he needs to generate random permutation of integers from 1 to n.

Unfortunately, Peter doesn't know good algorithm to do so. So he uses the following algorithm. First initialize $a[i] = i$. Then t times choose uniformly random i from 1 to n and random j from 1 to n and swap $a[i]$ and $a[j]$.

John learned about Peter's method to generate a random permutation and decided to use his knowledge to win in the casino. John wonders what permutation occurs most frequently among permutations generated by Peter's method and what is its probability.

Help John to find that out.

Input

The input file contains several test cases.

Each test case consists of two integers on a line: n and t ($1 \le n \le 14$, $0 \le t \le 300$).

The last test case is followed by two zeroes that must not be processed.

Output

For each test case output two lines. The first line must contain n integers: the permutation that occurs most frequently. If there are several such permutations, output lexicographically smallest one. The second line must contain the probability of generating this permutation as an irreducible fraction p/q.

Examples

frequent.in	frequent.out
3 3 3 4 0 0	1 2 3 5/27 1 2 3 43/243

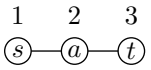
Problem G. Grid Wire Layout

Input file:	`grid.in`
Output file:	`grid.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The company *Grid Electronics* is developing layout for its new chip. The chip consists of nodes connected by wires. Each chip has one source node connected by a wire to exactly one other node, and one target node, similarly connected by a wire to exactly one other node. Nodes of a chip A with n nodes are numbered by integers from 1 to n, source node has number 1, target node has number n.

The structure of the chip can be described by a sequence of the following operations.

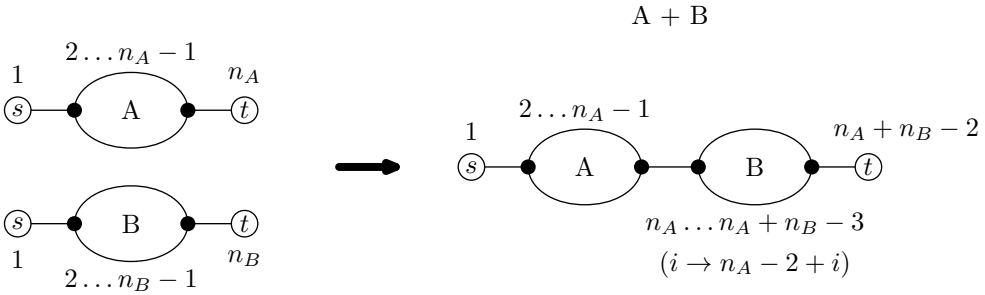
An atomic chip G_0 consists of source node s, operating node a and target node t, s is connected by wire to a, and a is connected by wire to t. Nodes have the following numbers: s is 1, a is 2, t is 3.



If A and B are chips with n_A and n_B nodes respectively, there are two ways to combine A and B to get a new chip.

Series connection $A + B$ is created in the following way. Target node of A and source node or B are removed, and nodes to which they were connected are directly connected by a wire. Source node of the new chip is source node of A, target node of the new chip is target node of B.

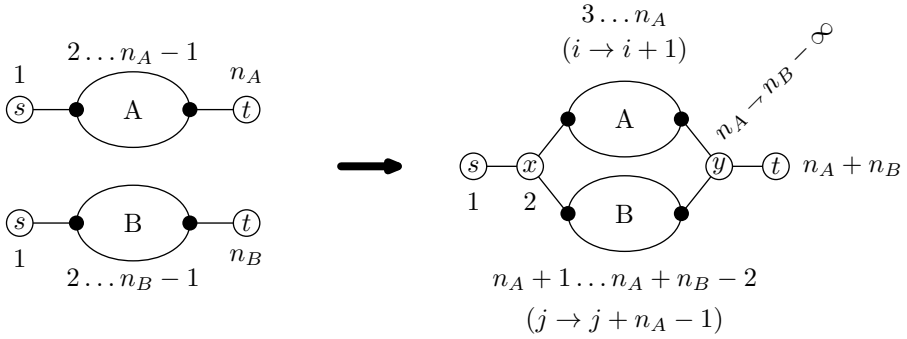
Nodes of the resulting chip are numbered from 1 to $n_A + n_B - 2$ in the following way. First all nodes of former chip A are numbered from 1 to $n_A - 1$ (target node is now removed). Then all nodes of former chip B are numbered from n_A to $n_A + n_B - 2$, if a node had number i in B it now has number $n_A - 2 + i$ (source node is now removed).



Parallel connection $A|B$ is created in the following way. Source nodes of A and B are merged to get a single node x connected to both nodes formerly connected to source nodes of A and B. Target nodes of A and B are merged in the same way to get a single node y. New source node s is created and connected to node x, new target node t is created and connected to node y.

Nodes of the resulting chip are numbered from 1 to $n_A + n_B$ in the following way. New source node gets number 1. Merged source nodes of A and B get number 2. Nodes of former chip A except source and target get numbers from 3 to n_A, node which had number i in A now gets number $i + 1$. Nodes of former chip B except source and target get numbers from $n_A + 1$ to $n_A + n_B - 2$, Node which had number j now gets number $j + n_A - 1$. Merged target nodes of A and B get number $n_A + n_B - 1$, and the new target node gets number $n_A + n_B$.

A | B



Grid Electronics has developed the new chip and now needs to create its layout on a rectangular board. Nodes correspond to points on it, and wires are segments that connect nodes. Wires must not intersect and may not pass through nodes other than their ends. Due to technology limitations, wire segments must be parallel to edges of the board. Such layout is called *grid layout*.

Help the company to design grid layout for its new chip.

Input

The input file contains several test cases.

Each test case starts with integer n ($0 \le n \le 100$) — number of instructions used to describe the chip. Let G_0 be atomic chip as described in the problem statement. The sequence $G_1, G_2, \ldots, G_n$ of chips is constructed. The following n lines contain instructions to construct the chips, the i-th of these lines describes the construction of G_i. The line contains either "+ a b" which means that G_i is isomorphic to $G_a + G_b$ or "| a b" which means that G_i is isomorphic to $G_a|G_b$ ($0 \le a, b < i$). The number of nodes in each chip doesn't exceed 1000. You have to make layout for G_n. Let us call it the final chip.

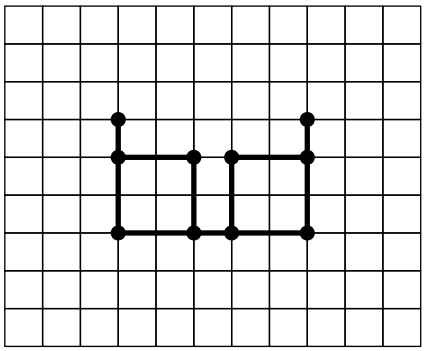
The last test case is followed by $n = -1$ which must not be processed.

The total number of nodes in all final chips in the input file doesn't exceed 100 000.

Output

For each test case print "−1" if there is no grid layout for its final chip. In the other case print the description of the layout.

The description must start with v — the number of nodes in the chip. It must be followed by v lines, the i-th line must contain coordinates of the i-th node. Edges of the board are parallel to coordinate axes. Coordinates must not exceed 10^9 by their absolute values.



Examples

grid.in	grid.out
2 | 0 0 + 1 1 0 -1	10 0 3 0 2 2 2 0 0 2 0 3 0 3 2 5 0 5 2 5 3 3 0 0 0 1 0 2

Problem H. Hentium Scheduling

Input file:	`hentium.in`
Output file:	`hentium.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

John is developing the new operating system Jindox Lion that will run on systems with new Hentium processor that has asymmetric dual-core support. There are two cores in Hentium, one core has fast memory access, another one has faster arithmetics. So scheduling memory-dependent processes on the first core and calculation intensive processes on the second one, one can take advantage of the architecture.

John is working on scheduling system boot processes on the two cores. There are n system initialization processes in Jindox. John has measured that the i-th process will take a_i nanoseconds to execute on the first core or b_i nanoseconds to execute on the second core. However, processes, scheduled at different cores are slower to communicate, so if the i-th process is scheduled at one core and the j-th process is scheduled at another one, it will take additional $c_{i,j}$ nanoseconds for them to communicate with each other.

Since John wants his system to be as stable as possible, he doesn't want processes at different cores to execute simultaneously. Therefore there is a special semaphore that ensures that only one process at one of the cores is executed at each moment of time. Now John needs to arrange some processes to the first core, and others to the second, so that the total time of their execution and communication was as small as possible. Help him to do it.

Input

The input file contains several test cases.

The first line of each test case contains n — the number of processes ($2 \le n \le 100$). The following n lines describe processes, each line contains two integers: a_i and b_i — execution times at the first core and at the second core, respectively ($1 \le a_i, b_i \le 10^9$). The following n lines contain n integers each, the j-th number of the i-th line is $c_{i,j}$ ($0 \le c_{i,j} \le 10^9$, $c_{i,i} = 0$, $c_{i,j} = c_{j,i}$).

Input is followed by $n = 0$, it must not be processed. The sum of n for all test cases doesn't exceed 1000.

Output

For each test case print two lines. The first line must contain one integer: the total time of execution and communication of processes in case of optimal scheduling, in nanoseconds.

The second line must contain n integers: for each process print 1 if it must be executed on the first core, or 2 if it must be executed on the second core. If there are several optimal solutions, output any one.

Examples

hentium.in	hentium.out
4 1 8 2 5 10 1 15 2 0 2 0 0 2 0 10 4 0 10 0 1 0 4 1 0 0	11 1 2 2 2

Problem I. IQ Test

Input file:	iq.in
Output file:	iq.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Ivan is working in Authorized Center of Mind Inspection and Certification for Practical Considerations (ACMICPC). Recently the center has received the new task of running IQ test for Important Bureaucracy Ministry (IBM) employees.

There are n IBM employees to be tested. All of them would come to ACMICPC at the beginning of the workday and then pass test in groups. Employees may be divided to groups arbitrarily, and groups may pass test in any order. For each employee the head of IBM has provided t_i — the time it would take him to complete the test. The time it takes to complete the test for a group is equal to the maximum time it takes to complete the test for its members.

Ivan would like to organize such groups that the total time spent by IBM employees in ACMICPC would be minimal possible. Let c_i denote the time when the group that the i-th employee is assigned to completes its test. Ivan would like to minimize $\sum_{i=1}^{n} c_i$.

Help Ivan to organize employees to groups.

Input

The input file contains several test cases.

Each test case starts with a line containing n — the number of employees ($1 \le n \le 10\,000$). The second line of each test case contains n integers: $t_1, t_2, \ldots, t_n$ — time needed by each employee to complete the test ($1 \le t_i \le 10^9$).

The last test case is followed by $n = 0$, it must not be processed.

The sum of n for all test cases in the input file is at most 10 000.

Output

For each test case output the following information. The first line of output must contain C — the optimal total time spent by IBM employees. The second line must contain g — the number of groups that the employees must be divided to. The following g lines must describe groups. Each group description must start with k_j — the number of persons in the group, followed by the persons it consists of. Persons are numbered from 1 to n. Groups must be described in order they should take test. If there are several optimal ways to assign employees to groups or order groups for testing, output any one.

Examples

iq.in	iq.out
4 1 2 7 8 0	24 2 2 1 2 2 3 4

In the given example the first group finishes test in 2 minutes, the second group finishes the test in 8 minutes. Employees 1 and 2 spend 2 minutes in ACMICPC, employees 3 and 4 spend 10 minutes (2 minutes while waiting for group 1 to complete the test, and 8 minutes to complete the test with their group).

Problem J. Jubilee Decoration

Input file:	jubilee.in
Output file:	jubilee.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Flatland has $n+1$ cities, one of the cities is its capital and other n cities are located around the country. The cities are traditionally numbered from 0 to n, the capital has number 0, other cities are numbered from 1 to n counterclockwise around the capital.

Government in Flatland is very centric, so most financial and cultural resources are concentrated in the capital. Thus all cities are connected to capital by roads. However, when traffic jams caused by giant trucks at capital roads started to annoy citizens, the Great Ring Road of n segments was constructed connecting all non-capital cities to a single cycle.

Therefore now there are $2n$ roads in Flatland, one between capital and city i for each i from 1 to n, one between city j and city $j+1$ for each j from 1 to $n-1$ and one between cities 1 and n.

This year president of Flatland has jubilee — 30 years as president. To celebrate this event, he decided to decorate all roads. He decided that he would use several types of decoration for roads, and he wanted citizens of his country to see various types as they travel.

President wants to choose k types of decorations for some k and choose decoration type for each road. After that for any pair of cities i and j (i and j from 0 to n) it must be possible to travel along roads from i to j in such way that no two roads on the path had the same decoration type.

Since developing new decoration type is rather expensive, president asked his Department of Jubilee and Transportation to create decoration plan that uses minimal possible number of decoration types. Help them to create the plan.

Input

The input file contains several test cases. Each test case consists of a single number n on a line by itself ($3 \le n \le 300$). The last line of the input file contains $n = 0$, it must not be processed.

Output

For each test case in the input file print three lines.

The first line must contain k — the minimal number of decoration types that must be used.

The second line must contain n integers from 1 to k each, the i-th of them must be the decoration type of the road from capital to city i.

The third line must contain n integers from 1 to k each, the i-th of them must be the decoration type of the road from city i to city ($i \bmod n + 1$).

Examples

jubilee.in	jubilee.out
4 0	2 1 1 2 2 1 2 1 2

Problem K. Kingdom Division 2

Input file:	`kingdom.in`
Output file:	`kingdom.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Once upon a time there was a kingdom ruled by a wise king. After forty years of his reign, by means of successful military actions and skillful diplomacy, the kingdom became a convex polygon with n vertices. This form of the kingdom greatly simplified tax collection and land division between the inhabitants, so everybody was happy.

But nothing lasts forever, and one autumn evening the king suddenly died. Due to the laws of the kingdom, the oldest son of the king should have taken the kingdom, but the problem was that he had two twin sons. It was widely known that the king loved his sons equally, so it was not even pronounced to choose only one son to rule.

After ten days of serious thoughts it was finally decided to split the kingdom into two parts — one per each son. Remembering the sad story of another kingdom that tried to divide the kingdom into two parts of equal area, wise men of the kingdom in this problem decided to use another way of kingdom division.

There are two cities in the kingdom, Andrewville and Bettyburg. It was decided to divide the kingdom by a straight line in such way that:

- The line connects two vertices of the polygon.
- Andrewville and Bettyburg are strictly inside of the parts that the polygon is divided to, and belong to different parts.

However it turned out that there are several ways to do the division. People of the kingdom don't know how to choose the optimal way to do it, so they first decided to count the number of ways. Help them to do it.

Input

The input file contains several test cases. Each test case starts with n — the number of vertices of the polygon ($4 \le n \le 100\,000$). The following n lines contain vertices of the polygon in counterclockwise order. No three vertices are on the same line.

Two lines follow, containing coordinates of Andrewville and Bettyburg, respectively. The cities are at different points strictly inside the given polygon.

Coordinates of all points do not exceed 10^9 by their absolute values.

The last test case is followed by $n = 0$, it must not be processed.

The sum of n for all test cases in the input file is at most $100\,000$.

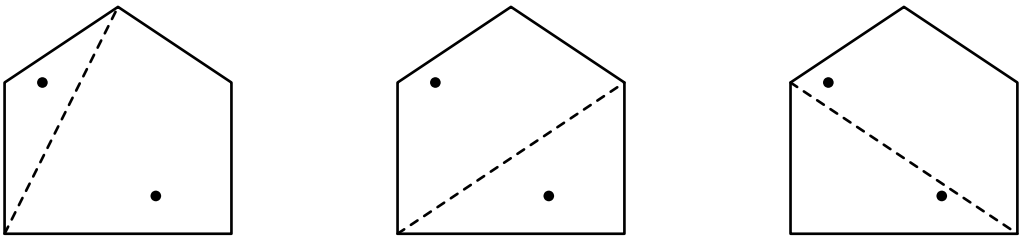
Output

For each test case output one integer — the number of ways to divide the kingdom.

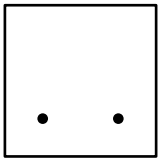
Examples

kingdom.in	kingdom.out
5 0 0 6 0 6 4 3 6 0 4 4 4 1 1 4 4 0 0 4 0 4 4 0 4 1 1 3 1 0	3 0

The picture below shows three ways to divide the kingdom in the first example.



There is no way to divide the kingdom in the second example.



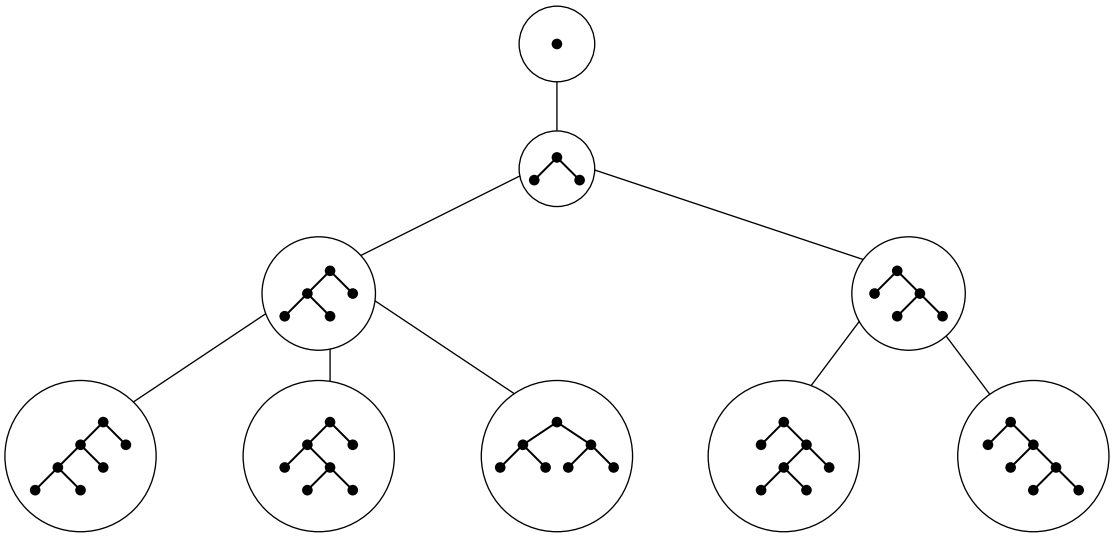
Problem L. Least Common Ancestor

Input file:	`lca.in`
Output file:	`lca.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

Least common ancestor is a well known problem for trees. In this problem you have to implement least common ancestor algorithm in a tree composed of complete rooted binary trees as vertices.

In a complete rooted binary tree each node either has exactly two children: left and right (internal node), or has no children (leaf).

Let us introduce the following parent relation on complete rooted binary trees. Consider a tree T. Find x — the rightmost internal vertex both children of which are leafs. Then parent of T is the tree which is obtained from T be removing children of x. A single-vertex tree obviously is the only one that has no parent. This relation allows to organize all complete rooted binary trees to a single infinite tree R, the top four levels of which are shown on a picture below.



You are given two trees T_1 and T_2. Find a tree T_3 which is a least common ancestor of T_1 and T_2 in R.

Input

The input file contains several test cases.

Each test case contains description of two complete binary rooted trees. Each tree description starts with n — the number of vertices in it ($1 \le n \le 149\,999$, n is odd). Let us number vertices of the tree from 1 to n in such way that parent of vertex x has number which is strictly smaller than x. The following n lines describe tree structure: the i-th of these lines contains numbers of left and right child of vertex i, correspondingly, or 2 zeroes if vertex i is a leaf.

The last test case is followed by a line that contains zero. It must not be processed. The total number of vertices in all trees in the input doesn't exceed 300 000.

Output

For each test case output a tree that is least common ancestor of the trees in the input. Use the same format as in the input file.

Examples

lca.in	lca.out
7 2 3 4 5 0 0 6 7 0 0 0 0 0 0 7 2 3 4 5 6 7 0 0 0 0 0 0 0 0 7 2 3 4 5 0 0 6 7 0 0 0 0 0 0 7 2 3 0 0 4 5 6 7 0 0 0 0 0 0 0	5 2 5 3 4 0 0 0 0 0 0 3 2 3 0 0 0 0