Problem A. Finite Automata

Input file:	`automata.in`
Output file:	`automata.out`
Time limit:	1 second
Memory limit:	64 megabytes

The *Broken Tiles* company has designed a robot for tiling the roads. The robot has an infinite supply of pavement tiles, each of which has the size of 2×1 feet. The roads to tile have a size of $m \times n$ feet. Here m is the width of the road, and n is the length of the road.

The program for the robot is the sequence of the commands. There are three commands: '`H`', '`V`' and '`S`'. The robot considers the road to tile consisting of $m \times n$ unit squares. The road runs from west to east.

The robot begins the execution of the program standing in the north-western square of the road. Each step it considers the current command. If it is '`H`', the robot puts the tile horizontally — the longer side from east to west, the western square of the tile occupies the current square. If the command is '`V`', the robot puts the tile vertically — the longer side from north to south, the northern square of the tile occupies the current square. Finally, if the command is '`S`', the robot does nothing on the current square. After executing the command, the robot moves one square southwards. If it crosses the border of the road, it moves one square eastwards, and moves to the northern square of the new column.

The road is said to be tiled correctly, if all of its squares are covered by the tiles, and no tiles overlap each other or cross the border of the road. The program is said to be correct for m, if it causes the robot to tile the $m \times n$ road correctly for some n, and after finishing the execution of the program the robot steps out of the southeastern corner of the road. For example, the program "`HHSS`" is correct for 2 (it correctly tiles the road for $n = 2$). The program "`HHV`", in turn, is incorrect for 2 for two reasons: first two tiles overlap with the third tile, and the robot does not leave the road in the end of the program for any n.

The designers of the company asked the main programmer of the company to write the program that would verify the correctness of the program for the robot.

But the main programmer of the company has recently learned the theory of finite automata and decided that everything should be programmed using finite automata only. Fortunately, it turned out, that for each m there indeed exists a finite automaton that accepts those and only those strings that form a correct for m program for the robot.

But the main programmer has gone to the Open Automata Documentation summit, leaving you alone with his ideas. The designers (and, more important, managers) are waiting for the verification automaton. So given m, you have to create a finite automaton for recognizing the correct programs. Since this is a business project, the automaton must be deterministic.

Recall, that the deterministic finite automaton (DFA) is an ordered set $\langle \Sigma, U, s, T, \varphi \rangle$ where Σ is the finite set called *input alphabet* ($\Sigma = \{$`H`, `V`, `S`$\}$ in our case), U is the finite set of *states*, $s \in U$ is the *initial state*, $T \subset U$ is the set of *terminal states* and $\varphi : U \times \Sigma \to U$ is the *transition function*.

The input of the automaton is the string α over Σ. Initially the automaton is in state s. Each step it reads the first character c of the input string and changes its state to $\varphi(u, c)$ where u is the current state. After that the first character of the input string is removed and the step repeats. If the automaton is in the terminal state after its input string is empty, it accepts the initial string α, in the other case it rejects it.

Input

The input file contains m ($1 \le m \le 10$).

Output

The first line of the output file must contain u — the number of states of the automaton, and s — the

initial state (states are numbered from 1 to u). The number of states must not exceed 20 000.

The second line must contain t — the number of terminal states, followed by t integer numbers — the terminal states themselves.

The following u lines must contain three numbers each — the i-th of these lines must contain $\varphi(i, H)$, $\varphi(i, V)$, and $\varphi(i, S)$.

Example

automata.in	automata.out
2	5 1 1 1 2 3 4 5 4 4 4 4 1 4 4 4 4 4 3

Problem B. Bacteria

Input file:	bacteria.in
Output file:	bacteria.out
Time limit:	1 second
Memory limit:	64 megabytes

One biological laboratory has recently completed an experiment of planting two new cultures of bacteria. The problem is that the cultures cannot live without each other, so they only exist together. However, to analyze the properties of each culture by itself, the scientists learned the way to plant bacteria with various density, so the ratio of the cultures in different parts of the experiment plate is different.

That is, let the experiment plate has the form of a unit square with corner coordinates $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$. The small $dx \times dy$ part of the plate with coordinates (x, y) contains a number of first culture bacteria proportional to $(\alpha x + \beta)\, dx\, dy$, and a number of second culture bacteria proportional to $(\gamma y + \delta)\, dx\, dy$.

Now the scientists want to send the results of the experiment to two friendly labaratories. To do it, they want to separate the experiment plate with the bacteria on it to two parts by a straight line. They want the line separate both bacteria of the first and of the second culture to the equal parts (bacteria are so small, that we consider them continuously covering the plate with the density described). Help them to do so.

Input

Input file contains four integer numbers: α, β, γ, and δ ($0 \le \alpha, \beta, \gamma, \delta \le 100$).

Output

Output three real numbers: a, b and c such that the line $ax + by + c = 0$ is the required one. If there is no solution, output three zeroes.

Example

bacteria.in	bacteria.out
1 0 1 0	0.707106781186548 0.7071067811865471 -0.8260806096503981

Problem C. Express Trains

Input file:	`express.in`
Output file:	`express.out`
Time limit:	1 second
Memory limit:	64 megabytes

The ministry of transport of Flatland has decided to build the new system of experss railways for the special extremely fast trains. But the budget of the country is limited because of the high course of the fleugric, the Flatland national currency. Therefore the ministry was forced to put some strong restrictions on the system of the railways.

After a discussion it was decided that the railways must be organized in such a way that there were no need for the complicated control system. That is, the railways must be running from west to east, without forks. Each railway must connect two cities, and may pass through some intermediate cities. Since the citizens of the country are concerned about its ecology, there must be at most one railway passing through each city, including the cities at the ends of the railway. Since there is not enough money to make a special infrastructure for the railways, the railways must go along existing roads.

After further inspection it was found out that the funds allocated for the project are only enough to build two railways. The minister has decided that the most important task at the moment is to connect the capital of the country A to the city B where the main Flatland university is located, and the city C where the big machine plant is operating to the main country port D. Thus, the two railways must run from west to east and connect A with B, and C with D, respectively.

Help the minister to find the way to build the railways.

Input

The first line of the input file contains n and m — the number of the cities in Flatland, and the number of existing roads ($4 \le n \le 2000$, $0 \le m \le 100\,000$). The second line contains four different numbers — the numbers of cities A, B, C and D, respectively.

Let cities be numbered from west to east. No two cities in Flatland are on the same longitude. The following m lines describe roads. Each line contains two numbers — a_i and b_i — the cities connected by the corresponding road ($a_i < b_i$). There can be several roads between a pair of cities.

Flatland is quite small, so there is no cycle of roads that goes around the planet.

Output

On the first line of the output file print "`YES`", if it is possible to build two railways that satisfy all the conditions described, and "`NO`" in the other case.

If it is possible to build the railways, the following two lines must contain the descriptions of the railways. Each description must be a sequence of the cities the railway goes through, from west to east.

Example

express.in	express.out
6 8 1 6 5 2 1 2 2 3 1 3 2 4 4 5 4 6 3 6 3 4	YES 1 3 6 2 4 5
5 4 1 2 4 5 1 3 2 3 3 4 3 5	NO

Problem D. Merge Sort

Input file:	`merge.in`
Output file:	`merge.out`
Time limit:	1 second
Memory limit:	64 megabytes

Merge sort is a famous sorting algorithm that applies divide-and-conquer approach to sorting. The algorithm proceeds as follows.

The algorithm is recursive, it sorts the segment of the initial array. The parameters of the recursive procedure are the indices l and r of the left and the right element of the segment to be sorted. If there is only one number to sort, the algorithm does nothing. In the other case, the array is divided into two segments: from l to m and from $m+1$ to r, where $m = \lfloor (l+r)/2 \rfloor$, and both parts are sorted recursively.

After that the sorted parts are merged. Two indices are used, one for the first of the sorted segments, the other for the second one. Initially the indices point to the leading elements of their segments. Each step the two pointed elements are compared. The smaller one is put to the end of the temporary array, and the corresponding index is increased by one. When one of the indices leaves its segment, the rest of the other segment is directly copied to the end of the temporary array. After that the elements are moved back from the temporary array to the original one. The segment is now sorted.

To sort the whole array of n elements, the algorithm is called with parameters 1 and n.

Note, that the number of comparisons made by the merge sort algorithm depends on the original array. For example, let at some step of the algorithm to segments of total length k be merged. If the elements of the first segment are all smaller than the elements of the second one, only about $k/2$ comparisons are made, but if the elements are mixed up, there can be up to $k-1$ comparisons.

In this problem we consider sorting of arrays containing n different integer numbers ranging from 1 to n. You have to determine the maximal possible number of comparisons needed by the merge sort algorithm for some array, find the lexicographically smallest such array, and the number of such arrays.

Array $(a_1, a_2, \ldots, a_n)$ is lexicographically smaller than array $(b_1, b_2, \ldots b_n)$ if there exists i such that $a_1 = b_1$, $a_2 = b_2$, $\ldots$, $a_{i-1} = b_{i-1}$, $a_i < b_i$.

Input

The input file contains integer number n ($1 \le n \le 200$).

Output

On the first line of the output file print the maximal number of comparisons needed to sort the array of n elements using merge sort.

On the second line of the output file print the lexicographically smallest such array.

On the third line of the output file print the number of arrays that require the maximal number of comparisons to sort.

Example

merge.in	merge.out
4	5 1 3 2 4 16
5	8 1 2 4 3 5 48

Problem E. Guarding the Place of the Murder

Input file:	`murder.in`
Output file:	`murder.out`
Time limit:	1 second
Memory limit:	64 megabytes

The main action in a new movie of a famous director and producer Geo Lookass takes place in one ancient Greece temple. The friend of the main hero is found murdered inside the temple.

The police coming to the place of the murder needs to surround it with a special tape and guard it. The temple has the form of a dome supported by a number of columns. So, the most natural way to surround the temple is to attach the tape to the columns.

Now, planning the budget of the movie, Geo wonders, what is the minimal length of the tape needed to surround the temple. Help him to determine that.

Input

The first line of the input file contains n — the number of the columns ($1 \le n \le 500$). The following n lines describe the columns — each line contains the coordinates of the center, and the radius of the column.

All coordinates are integer and do not exceed 5 000 by their absolute value. Radii are positive integer numbers and do not exceed 5 000. No two columns intersect or touch each other.

Output

Output the minimal length of the tape needed to surround the temple. Your answer must be accurate up to 10^{-4}.

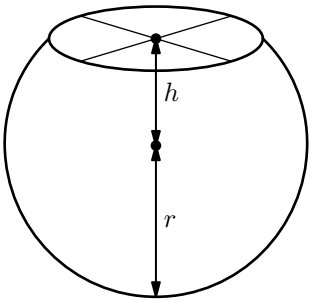
Example

murder.in	murder.out
4 0 0 1 3 3 1 3 0 1 0 3 1	18.2831853071795865

Problem F. Wall Painting

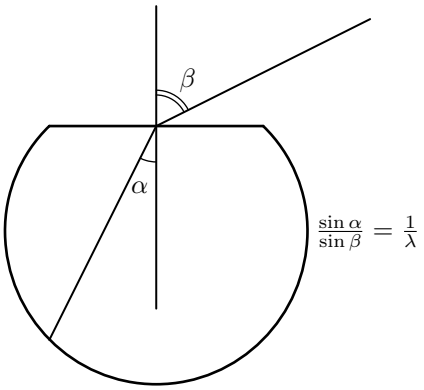
Input file:	painting.in
Output file:	painting.out
Time limit:	1 second
Memory limit:	64 megabytes

Recently archeologists have found a new strange artefact in South America. Their find is a vessel that has the form of the part of the sphere filled with some strange liquid substance. The inner walls of the vessel are painted with some interesting pictures.



However, because the liquid inside the vessel has a high index of refraction, the part of the inner wall cannot be seen from the outside. Now the archeologists wonder, what part of the wall they can see from outside. Help them to find that out.

For the purpose of this problem, the index of refraction of air is 1.0.



Input

The input file contains two real numbers: the value of h/r, and the index of refraction λ of the liquid inside the vessel ($0 < h/r < 1$, $1 < \lambda \leq 10$).

Output

Output one real number — the part of the inner surface of the vessel that can be seen from the outside. Your answer must be accurate up to 10^{-6}.

Example

painting.in	painting.out
0.6 2.0	0.8705127019

Problem G. Palindromes

Input file:	palindromes.in
Output file:	palindromes.out
Time limit:	1 second
Memory limit:	64 megabytes

A non-empty string is called a *palindrome* if it coincides with its reverse, i.e. it the same if read from left to right, and from right to left. The examples of the palindromes are: "ABBA", "MADAMIMADAM", "ABACABADABACABA".

Given a string s, find the number of different palindromes that are substrings of s.

Input

The input file contains a string s that consists of 1 to 5 000 of capital English letters.

Output

Output the number of different palindromes that are substrings of s.

Example

palindromes.in	palindromes.out
ABACABADABACABA	15

Problem H. Prime Sum

Input file:	`prime.in`
Output file:	`prime.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Let us consider a representation of a positive integer number n as a sum of one or more integer numbers:

$$n = x_1 + x_2 + \ldots + x_k.$$

Let us call such sum *prime*, if all terms in it are pairwise relatively prime. Recall, that x and y are called relatively prime, if their greatest common divisor is 1.

Given n, find the number of ways it can be represented as a prime sum. The ways that differ only by the order of the terms are considered the same. For example, there are six such representation for $n = 5$:

$$5 = 5$$
$$5 = 4 + 1$$
$$5 = 3 + 2$$
$$5 = 3 + 1 + 1$$
$$5 = 2 + 1 + 1 + 1$$
$$5 = 1 + 1 + 1 + 1 + 1$$

Input

Input file contains one number n ($3 \le n \le 150$).

Output

Print one number — the number of ways n can be represented as a prime sum.

Example

prime.in	prime.out
5	6

Problem I. Sharing the Sweets

Input file:	`sweets.in`
Output file:	`sweets.out`
Time limit:	1 second
Memory limit:	64 megabytes

The parents have made Peter a birthday present — a pack of n sweets. But Peter doesn't like sweets. He has invited k friends to his birthday and wants to share the sweets among them. To do it, he would like to split his sweets to k non-empty parts.

But the boy is confused — there are so many ways to do it. For example, if $n = 6$ and $k = 3$, there are 3 ways to divide the sweets — make a part of 4 sweets, and two parts of 1 sweet; make a part of 3 sweets, a part of 2 sweets, and a part of 1 sweet; or just make three equal parts of 2 sweets.

Help Peter, find the number of ways to divide his n sweets to k parts.

Input

Input file contains two integer numbers — n and k ($1 \le k \le n \le 1500$).

Output

Output one number — the number of ways to divide n sweets to k parts.

Example

sweets.in	sweets.out
6 3	3

Problem J. Tree Analysis

Input file:	`tree.in`
Output file:	`tree.out`
Time limit:	1 second
Memory limit:	64 megabytes

A *rooted tree* is a directed graph with one selected node r, called *root*, that satisfied the following conditions:

1. all nodes are reachable from r;
2. r has no edges entering it;
3. each node u except r has exactly one edge entering it, the source of this edge is called the *parent* of u.

Each node u in a rooted tree defines a set $T(u)$ of nodes, reachable from it. The graph induced by $T(u)$ is also a rooted tree, it is called a subtree rooted in u.

Two graphs G_1 and G_2 are called *isomorphic* if there is a bijection φ between their sets of vertices, such that $u \to v$ is an edge in G_1 if and only if $\varphi(u) \to \varphi(v)$ is an edge in G_2. Two rooted trees are isomorphic if they are isomorphic as graphs.

One way to check whether the two trees are isomorphic uses *subtree isomorphism relation*. Two nodes u and v of the tree are called *subtree isomorphic* if subtrees rooted in u and v respectively are isomorphic. Clearly, subtree isomorphism relation is an equivalence relation. Therefore, all nodes of the tree can be divided into equivalence classes by this relation.

Given a rooted tree, find its subtree isomorphism relation equivalence classes.

Input

The first line of the input file contains n — the number of nodes in the tree ($1 \le n \le 100\,000$). Let nodes be numbered in such a way that the root has the number 1 and for each other vertex its number is greater than the number of its parent.

The rest of the file contains $n - 1$ integer numbers separated by spaces and line feeds. The i-th of these numbers is the parent of the $(i + 1)$-th node.

Output

Output n integer numbers — for each node output some integer number ranging from 1 to n. The numbers for any two nodes must be equal if and only if these nodes are subtree isomorphic.

Example

tree.in	tree.out
9 1 2 2 1 5 5 7 7	1 2 3 3 4 3 2 3 3