Problem A. Area 51

Input file: `area.in`
Output file: `area.out`

Michael and Nick are living near the famous top-secret "Area 51" facility. The facility is enclosed by a fence and is so large that for the purpose of this problem we consider the fence being a line that stretches infinitely into both directions.

Only extremely brave boys are not scared to go to the fence and peek at the facility. Nick is among the brave ones. He once came to the fence and saw a number of chimneys with distinct shapes. As a proof of his bravery he tells everybody what chimneys he saw from his left to his right.

Michael's father is working at "Area 51" and has a facility's map at his home. Michael found this map and he can now verify Nick's claim of being near the facility's fence. However, it turns out to be complicated, and your task is to write a program to perform this verification.

On a map distinctly shaped chimneys are denoted by capital letters from A to Z. Each letter denotes a distinct shape, but chimneys with this shape can appear more than once on a map. The map uses Cartesian coordinate system oriented so that the fence is Ox axis and all chimneys are located on a half-plane with a positive y coordinate. All chimneys are considered to be points (their sizes and actual geometrical shapes are ignored for the purpose of this problem).

Nick claims that he looked from a point on the fence where no two chimneys were on the same line of his sight (a line that originates from his point of view). It means that at the point he looked from, all the chimneys he saw had a well-defined order from left to right.

Michael have already made a preliminary verification of Nick's claim. He made sure that the number of distinctly shaped chimneys matches their number on the map. Now Michael needs to perform a final verification — to get a list of x coordinates on a fence (if any) where the corresponding arrangement of chimneys could be seen from. This information shall be presented as an ordered list of open intervals (a_1, b_1), (a_2, b_2), ..., (a_n, b_n), so that $a_1 < b_1 \leq a_2 < b_2 \leq ... \leq a_n < b_n$. Asterisk symbol ("$*$") is used in place of a_1 and/or b_n to denote interval that extends to infinity on the left or on the right correspondingly. Note, that $b_i = a_{i+1} = x$ in case where Nick could not have been at the point x on a fence, because he would have seen more than one chimney on a single line of his sight, but being to the left or to the right of x yields the order of chimneys that he saw.

Input

The first line of the input file contains an integer number m — the number of chimneys at the "Area 51" facility ($1 \leq m \leq 100$). The second line of the input file contains a string of m letters from A to Z that describe the chimneys that Nick saw from his left to his right. A single letter can be used more than once (if Nick saw the same shape more than once). Then follow m lines that describe chimneys on the map. Each line contains three tokens separated by spaces — chimney shape letter (from A to Z), and two integers x_i and y_i — chimney coordinates ($-100 \leq x_i \leq 100$, $0 < y_i \leq 100$). On these m lines letters appear in arbitrary order, but each letter from A to Z appears the same number of times as on the second line of the input file. No two chimneys have the same coordinates.

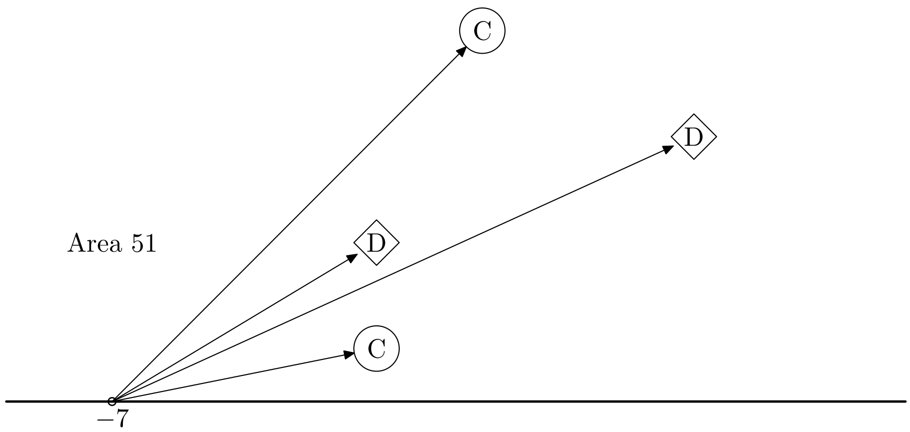
Output

Write to the first line of the output file a single integer number n — the number of intervals that describe x coordinates on a fence where Nick could have seen the corresponding arrangement of chimneys from. Write to the second line of the output file $2n$ numbers a_1, b_1, ... , a_n, b_n using "$*$" instead of a number a_1 and/or b_n to denote infinity. Numbers must be precise up to 10^{-6}.

Sample input and output

area.in	area.out
4 CDDC C 0 7 D 4 5 C -2 1 D -2 3	3 * -11 -11 -3.5 14 *
4 DCCD C 0 7 D 4 5 C -2 1 D -2 3	2 -3.5 -2.333333 -2.333333 -2
4 DCDC C 0 7 D 4 5 C -2 1 D -2 3	0

The picture below shows that if the boy looks from the point $x = -7$ he sees the chimneys in the following order: C, D, D, C. It is so for any point from the set $(-\infty, -11) \cup (-11, -3.5) \cup (14, +\infty)$ — the first example from the problem statement.



Problem B. Brackets Removal

Input file: `brackets.in`
Output file: `brackets.out`

Let us consider arithmetic expressions that consist of variables denoted by lower-case letters "a" to "z"; four *binary* arithmetic operations: addition ("+"), subtraction ("−"), multiplication ("∗"), and division ("/"); opening ("(") and closing (")") round brackets. The normal order of precedence is used — multiplication and division have the highest precedence, addition and subtraction have the lowest precedence. Operations of the same precedence are evaluated from left to right (for example $a - b + c = (a - b) + c$).

Thus, the grammar for the expressions is the following:

$$\begin{array}{rcl} \langle\text{expression}\rangle & \longrightarrow & \langle\text{term}\rangle \mid \langle\text{expression}\rangle + \langle\text{term}\rangle \mid \langle\text{expression}\rangle - \langle\text{term}\rangle \\ \langle\text{term}\rangle & \longrightarrow & \langle\text{factor}\rangle \mid \langle\text{term}\rangle * \langle\text{factor}\rangle \mid \langle\text{term}\rangle \,/\, \langle\text{factor}\rangle \\ \langle\text{factor}\rangle & \longrightarrow & \langle\text{variable}\rangle \mid (\langle\text{expression}\rangle) \\ \langle\text{variable}\rangle & \longrightarrow & a \mid b \mid \ldots \mid z \end{array}$$

Your task is to rewrite the given expression so that its semantics is not changed, but the resulting expression has the minimal number of round brackets.

You can remove any excessive brackets that do not change the order of evaluation, for example

$$(a + b) + (c) \Rightarrow a + b + c,$$

$$(a * b)/(c) \Rightarrow a * b/c,$$

and you can rewrite expressions using the following rules:

- If A and B are arbitrary expressions, you can change $A + (B)$ to $A + B$, for example

$$a - g/h + (b + c - d + e * (f + h - i)) \Rightarrow a - g/h + b + c - d + e * (f + h - i).$$

- If A and B are arbitrary expressions, you can change $A - (B)$ to $A - B'$ where B' is obtained from B by replacing all top-level "+" operations to "−" operations and vice versa, for example

$$a - g/h - (b + c - d + e * (f + h - i)) \Rightarrow a - g/h - b - c + d - e * (f + h - i)).$$

- If A and B are arbitrary terms, you can change $A * (B)$ to $A * B$, for example

$$x/(y + z) * (a * (b - c)/d/(e/f)) \Rightarrow x/(y + z) * a * (b - c)/d/(e/f).$$

- If A and B are arbitrary terms, you can change $A/(B)$ to A/B', where B' is obtained from B by replacing all top-level "∗" operations to "/" operations and vice versa, for example

$$x/(y + z)/(a * (b - c)/d/(e/f)) \Rightarrow x/(y + z)/a/(b - c) * d * (e/f).$$

You can think about these transformations as ones that only use "+" and "∗" associativity, the fact that "−" is the reverse operation to "+", "/" is the reverse operation to "∗", and nothing else.

You can apply the described transformations and remove excessive brackets as many times as you need to get the expression with the minimal number of round brackets.

Input

The input file contains a single line with the expression. Expression does not have any leading, trailing, or inner spaces and consists of at most 1000 characters.

Output

Write to the output file a single line with the same expression that is rewritten with the minimal number of round brackets. Do not write any spaces.

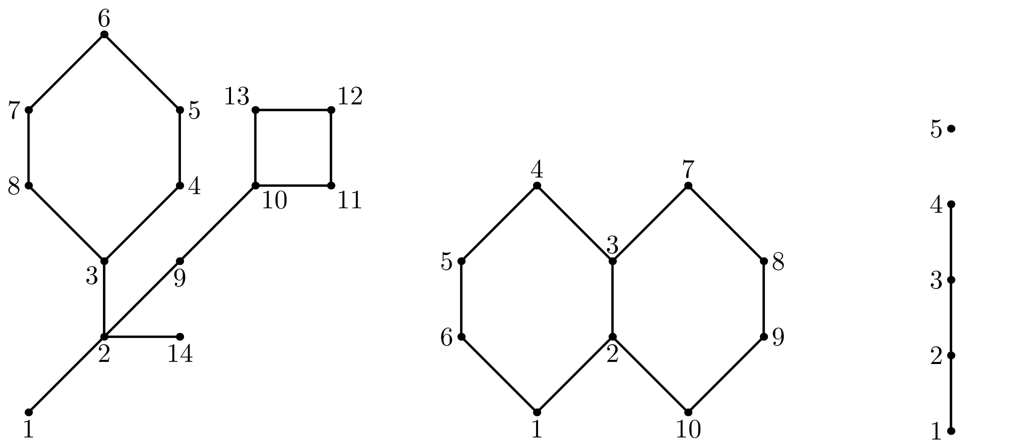
Sample input and output

brackets.in
`((a-b)-(c-d)-(z*z*g/f)/(p*(t))*((y-u)))`
brackets.out
`a-b-c+d-z*z*g/f/p/t*(y-u)`

Problem C. Cactus

Input file: `cactus.in`
Output file: `cactus.out`

Cactus is a connected undirected graph in which every edge lies on at most one simple cycle. Intuitively cactus is a generalization of a tree where some cycles are allowed. Your task first is to verify if the given graph is a cactus or not. Important difference between a cactus and a tree is that a cactus can have a number of spanning subgraphs that are also cactuses. The number of such subgraphs (including the graph itself) determines *cactusness* of a graph (this number is one for a cactus that is just a tree). The cactusness of a graph that is not a cactus is considered to be zero.



A cactus | Not a cactus | Not a cactus

The first graph on the picture is a cactus with cactusness 35. The second graph is not a cactus because edge $(2, 3)$ lies on two cycles. The third graph is not a cactus because it is not connected.

Input

The first line of the input file contains two integer numbers n and m ($1 \le n \le 20000$, $0 \le m \le 1000$). Here n is the number of vertices in the graph. Vertices are numbered from 1 to n. Edges of the graph are represented by a set of edge-distinct paths, where m is the number of such paths.

Each of the following m lines contains a path in the graph. A path starts with an integer number k_i ($2 \le k_i \le 1000$) followed by k_i integers from 1 to n. These k_i integers represent vertices of a path. Path can go to the same vertex multiple times, but every edge is traversed exactly once in the whole input file. There are no multiedges in the graph (there is at most one edge between any two vertices).

Output

Write to the output file a single integer number — the cactusness of the given graph. Note that cactusness can be quite a large number.

Sample input and output

`cactus.in`	`cactus.out`
14 3 9 1 2 3 4 5 6 7 8 3 7 2 9 10 11 12 13 10 2 2 14	35
10 2 7 1 2 3 4 5 6 1 6 3 7 8 9 10 2	0
5 1 4 1 2 3 4	0

Problem D. Double Patience

Input file: `double.in`
Output file: `double.out`

Double Patience is a single player game played with a standard 36-card deck. The cards are shuffled and laid down on a table in 9 piles of 4 cards each, faces up.

After the cards are laid down, the player makes turns. In a turn he can take top cards of the same rank from any two piles and remove them. If there are several possibilities, the player can choose any one. If all the cards are removed from the table, the player wins the game, if some cards are still on the table and there are no valid moves, the player loses.

George enjoys playing this patience. But when there are several possibilities to remove two cards, he doesn't know which one to choose. George doesn't want to think much, so in such case he just chooses a random pair from among the possible variants and removes it. George chooses among all possible pairs with equal probability.

For example, if the top cards are KS, KH, KD, 9H, 8S, 8D, 7C, 7D, and 6H, he removes any particular pair of (KS, KH), (KS, KD), (KH, KD), (8S, 8D), and (7C, 7D) with the equal probability of 1/5.

Once George's friend Andrew came to see him and noticed that he sometimes doesn't act optimally. George argued, that it is not important — the probability of winning any given patience with his strategy is large enough.

Help George to prove his statement — given the cards on the table in the beginning of the game, find out what is the probability of George winning the game if he acts as described.

Input

The input file contains nine lines. Each line contains the description of four cards that form a pile in the beginning of the game, from the bottom card to the top one.

Each card is described with two characters: one for rank, one for suit. Ranks are denoted as '`6`' for six, '`7`' for seven, '`8`' for eight, '`9`' for nine, '`T`' for ten, '`J`' for jack, '`Q`' for queen, '`K`' for king, and '`A`' for ace. Suits are denoted as '`S`' for spades, '`C`' for clubs, '`D`' for diamonds, and '`H`' for hearts. For example, "`KS`" denotes the king of spades.

Card descriptions are separated from each other by one space.

Output

Output one real number — the probability that George wins the game if he plays randomly. Your answer must be accurate up to 10^{-6}.

Sample input and output

double.in	double.out
AS 9S 6C KS JC QH AC KH 7S QD JD KD QS TS JS 9H 6D TD AD 8S QC TH KC 8D 8C 9D TC 7C 9C 7H JH 7D 8H 6S AH 6H	0.589314

Problem E. Exploring Pyramids

Input file: `exploring.in`
Output file: `exploring.out`

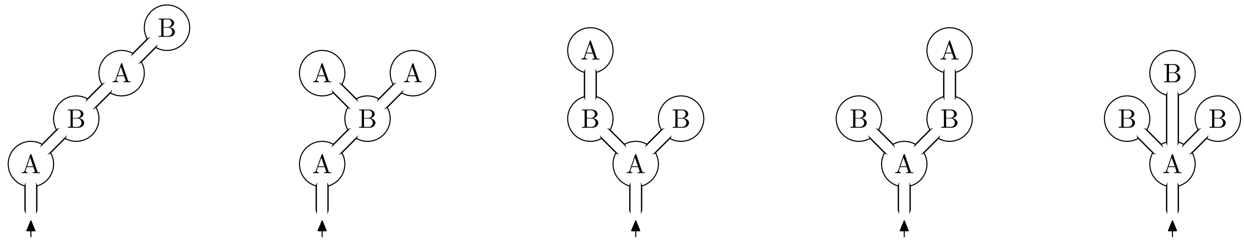
Archaeologists have discovered a new set of hidden caves in one of the Egyptian pyramids. The decryption of ancient hieroglyphs on the walls nearby showed that the caves structure is as follows. There are n caves in a pyramid, connected by narrow passages, one of the caves is connected by a passage to the outer world. The system of the passages is organized in such a way, that there is exactly one way to get from outside to each cave along passages. All caves are located in the basement of the pyramid, so we can consider them being located in the same plane. Passages do not intersect. Each cave has its walls colored in one of several various colors.

The scientists have decided to create a more detailed description of the caves, so they decided to use an exploring robot. The robot they are planning to use has two types of memory — the output tape, which is used for writing down the description of the caves, and the operating memory organized as a stack.

The robot first enters the cave connected to the outer world along the passage. When it travels along any passage for the first time, it puts its description on the top of its stack. When the robot enters any cave, it prints the color of its walls to its output tape. After that it chooses the leftmost passage among those that it has not yet travelled and goes along it. If there is no such passage, the robot takes the passage description from the top of its stack and travels along it in the reverse direction. The robot's task is over when it returns to the outside of the pyramid. It is easy to see that during its trip the robot visits each cave at least once and travels along each passage exactly once in each direction.

The scientists have sent the robot to its mission. After it returned they started to study the output tape. What a great disappointment they have had after they have understood that the output tape does not describe the cave system uniquely. Now they have a new problem — they want to know how many different cave systems could have produced the output tape they have. Help them to find that out.

Since the requested number can be quite large, you should output it modulo 1 000 000 000. Please note, that the absolute locations of the caves are not important, but their relative locations are important, so the caves (c) and (d) on the picture below are considered different.



(a) (b) (c) (d) (e)

Input

The input file contains the output tape that the archaeologists have. The output tape is the sequence of colors of caves in order the robot visited them. The colors are denoted by capital letters of the English alphabet. The length of the tape does not exceed 300 characters.

Output

Output one integer number — the number of different cave systems (modulo 1 000 000 000) that could produce the output tape.

Sample input and output

exploring.in	exploring.out
ABABABA	5
AB	0

Problem F. Feel Good

Input file:	`feelgood.in`
Output file:	`feelgood.out`

Bill is developing a new mathematical theory for human emotions. His recent investigations are dedicated to studying how good or bad days influent people's memories about some period of life.

A new idea Bill has recently developed assigns a non-negative integer value to each day of human life. Bill calls this value the *emotional value* of the day. The greater the emotional value is, the better the day was. Bill suggests that the value of some period of human life is proportional to the sum of the emotional values of the days in the given period, multiplied by the smallest emotional value of the day in it. This schema reflects that good on average period can be greatly spoiled by one very bad day.

Now Bill is planning to investigate his own life and find the period of his life that had the greatest value. Help him to do so.

Input

The first line of the input file contains n — the number of days of Bill's life he is planning to investigate ($1 \le n \le 100\,000$). The rest of the file contains n integer numbers $a_1, a_2, \ldots a_n$ ranging from 0 to 10^6 — the emotional values of the days. Numbers are separated by spaces and/or line breaks.

Output

On the first line of the output file print the greatest value of some period of Bill's life.

On the second line print two numbers l and r such that the period from l-th to r-th day of Bill's life (inclusive) has the greatest possible value. If there are multiple periods with the greatest possible value, then print any one of them.

Sample input and output

feelgood.in	feelgood.out
6 3 1 6 4 5 2	60 3 5

Problem G. Guards

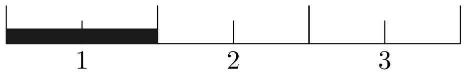
Input file: `guards.in`
Output file: `guards.out`

Agency of Criminal Matters (ACM) provides protection for corporate offices. The agency employs security guards that work in 12 hours shifts. Every 24 hours day consists of a 12 hours *daylight shift* and 12 hours *nighttime shift* (day starts with a daylight shift).

Security guards are hired to work on different schedules. Some schedules are simply periodic (certain pattern repeats in a specified number of days), while others are weekly (with a standard 7 days week) or depend on the day of the week. For this purpose Monday through Friday are considered to be *workdays*; Saturday and Sunday are considered to be *weekends*.

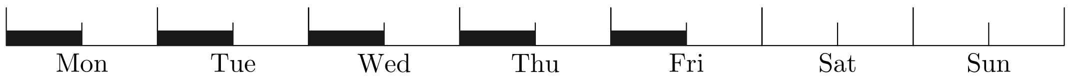
Each security guard works on one of the following four schedules:

1. Day of work (daylight and nighttime shifts) and two days (daylight and nighttime) of rest – work every third day.



Work on any day of week Monday through Sunday

2. Only daylight shifts on 5 workdays of a week (no work in nighttime and weekends).

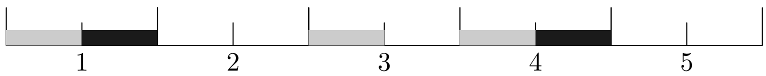


Work on the corresponding day of week

3. Day of work (daylight and nighttime), day of rest (daylight and nighttime), daylight shift of work, nighttime of rest, and one more day (daylight and nighttime) of rest — 3 shifts of work every 4 days.

Work on any day of week Monday through Sunday

4. Day of work (daylight and nighttime), day of rest (daylight and nighttime), day of work only during daylight (rest during nighttime), day of work (daylight and nighttime), day of rest (daylight and nighttime); but if any daylight shift falls on the weekend then it is cancelled — 3 daylight shifts and 2 nighttime shifts every 5 days with the exception of weekends (where only nighttime shifts are possible).



Work on any day of week Monday through Sunday
Work only on workdays — Monday through Friday

ACM has to provide protection for a location based on the following requirements. There has to be at least a certain number of guards during daylight shifts on workdays, at least a certain number of guards during daylight shifts on weekends, and at least a certain number of guards during nighttime shifts (it does not matter whether it is a workday or a weekend).

As an additional requirement (for simplicity of planning) the schedule of protection for every location has to be *regular*. In a regular schedule there is a fixed number of guards that work on any particular schedule during every daylight workday shift, nighttime workday shift, daylight weekend shift, and nighttime weekend shift. For example, if 4 guards on the 1st schedule work on some daylight workday shift, then 4 guards on the 1st schedule work on all daylight workday shifts (they might be different persons, though).

Your task is to determine the minimal number of guards that have to be hired for protection of the specific location given its requirements.

Input

The input file consists of a line with three integer numbers — n_1, n_2, and n_3 ($1 \le n_1, n_2, n_3 \le 1000$). Here n_1 is the minimum required number of guards during daylight shifts on workdays, n_2 is the minimum required number of guards during daylight shifts on weekends, and n_3 is the minimum required number of guards during nighttime shifts.

Output

Write to the output file a single line with four integer numbers — m_1, m_2, m_3, and m_4. Here m_i is the number of security guards working on i-th schedule that are needed to establish security of a location with the given requirements. You have to write an answer that requires minimal number of guards in total, choosing any answer among those.

Sample input and output

guards.in	guards.out
100 99 99	285 0 8 5
100 60 40	0 25 120 25

Problem H. Hardwood Cutting

Input file: `hardwood.in`
Output file: `hardwood.out`

John owns a furniture workshop. His clients are very rich people, so they often order furniture suites made of precious sorts of hardwood.

Recently John has got a series of orders from his clients, so now he needs to cut a hardwood board to several pieces. The board has a rectangular form of $m \times n$ feet. John has marked the outlines of the pieces to cut out on the board, and is planning to use his circular saw to cut it.

However, there is a little problem. Due to the construction of the circular saw, it is only possible to make straight cuts starting at the edge of the board. Although, after cutting away a part of the board John can take it away and make a cut from the new part of the edge, some pieces still cannot be separated using a circular saw. For example, pieces 'C' and 'D' on the picture below cannot be separated, neither can 'E' and 'Z'. To deal with such situations John will need to use his fret-saw to finish the cutting.

Now John wonders what is the maximal number of parts he can cut the board to with his circular saw, so that he needs less work to do with his fret-saw. Help him to find that out. After cutting some part away John can rearrange the parts in any way he likes.

Input

The first line of the input file contains two integer numbers m and n — the sizes of the board ($1 \leq m, n \leq 20$).

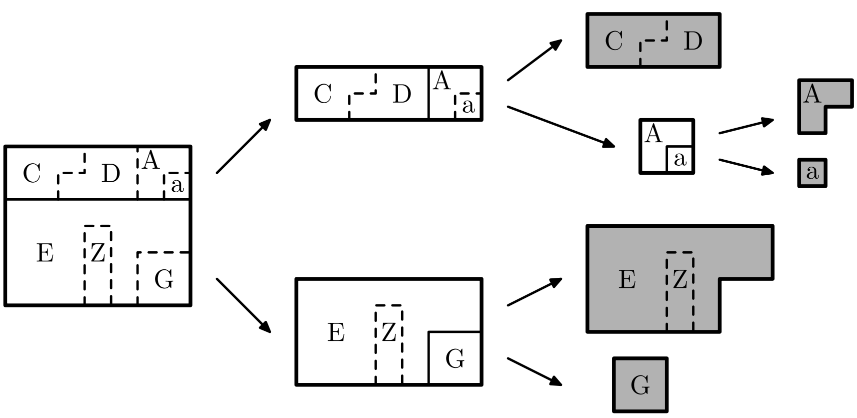
The following m lines contain n characters each and describe the marking of the board. Each unit square foot of the initial board is marked with some English letter or digit. Unit squares that belong to the same piece are marked with the same character. All unit squares that are marked with some character form an edge-connected piece of hardwood. Capital and lower-case letters are considered different.

Output

Output the number of parts that John can cut the board to with his circular saw.

Sample input and output

hardwood.in	hardwood.out
6 7 CCCDDAA CCDDDAa EEEEEEE EEEZEEE EEEZEGG EEEZEGG	5



Problem I. IP Networks

Input file: `ip.in`
Output file: `ip.out`

Alex is administrator of IP networks. His clients have a bunch of individual IP addresses and he decided to group all those IP addresses into the smallest possible IP network.

Each IP address is a 4-byte number that is written byte-by-byte in a decimal dot-separated notation "byte0.byte1.byte2.byte3" (quotes are added for clarity). Each byte is written as a decimal number from 0 to 255 (inclusive) without extra leading zeroes.

IP network is described by two 4-byte numbers — network address and network mask. Both network address and network mask are written in the same notation as IP addresses.

In order to understand the meaning of network address and network mask you have to consider their binary representation. Binary representation of IP address, network address, and network mask consists of 32 bits: 8 bits for byte0 (most significant to least significant), followed by 8 bits for byte1, followed by 8 bits for byte2, and followed by 8 bits for byte3.

IP network contains a range of 2^n IP addresses where $0 \le n \le 32$. Network mask always has $32 - n$ first bits set to one, and n last bits set to zero in its binary representation. Network address has arbitrary $32 - n$ first bits, and n last bits set to zero in its binary representation. IP network contains all IP addresses whose $32 - n$ first bits are equal to $32 - n$ first bits of network address with arbitrary n last bits. We say that one IP network is smaller than the other IP network if it contains fewer IP addresses.

For example, IP network with network address 194.85.160.176 and network mask 255.255.255.248 contains 8 IP addresses from 194.85.160.176 to 194.85.160.183 (inclusive).

Input

The first line of the input file contains a single integer number m ($1 \le m \le 1000$). The following m lines contain IP addresses, one address on a line. Each IP address may appear more than once in the input file.

Output

Write to the output file two lines that describe the smallest possible IP network that contains all IP addresses from the input file. Write network address on the first line and network mask on the second line.

Sample input and output

ip.in	ip.out
3 194.85.160.177 194.85.160.183 194.85.160.178	194.85.160.176 255.255.255.248

Problem J. Joseph's Problem

Input file:	`joseph.in`
Output file:	`joseph.out`

Joseph likes taking part in programming contests. His favorite problem is, of course, Joseph's problem. It is stated as follows.

> *There are n persons numbered from 0 to $n-1$ standing in a circle. The person number k, counting from the person number 0, is executed. After that the person number k of the remaining persons is executed, counting from the person after the last executed one. The process continues until only one person is left. This person is a survivor. The problem is, given n and k detect the survivor's number in the original circle.*

Of course, all of you know the way to solve this problem. The solution is very short, all you need is one cycle:

```
r := 0;
for i from 1 to n do
    r := (r + k) mod i;
return r;
```

Here "`x mod y`" is the remainder of the division of x by y

But Joseph is not very smart. He learned the algorithm, but did not learn the reasoning behind it. Thus he has forgotten the details of the algorithm and remembers the solution just approximately.

He told his friend Andrew about the problem, but claimed that the solution can be found using the following algorithm:

```
r := 0;
for i from 1 to n do
    r := r + (k mod i);
return r;
```

Of course, Andrew pointed out that Joseph was wrong. But calculating the function Joseph described is also very interesting.

Given n and k, find $\sum_{i=1}^{n}(k \bmod i)$.

Input

The input file contains n and k ($1 \le n, k \le 10^9$)

Output

Output the sum requested.

Sample input and output

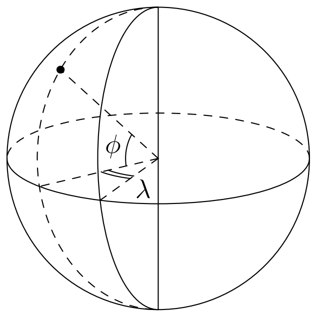
joseph.in	joseph.out
5 3	7

Problem K. Knockdown

Input file: `knockdown.in`
Output file: `knockdown.out`

The Evil Empire is devising a plan to destroy the world. This plan is called "Operation Knockdown". The plan is to place a number of high-yield nuclear bombs in different places around the world, so that their simultaneous detonation will destroy everything on the planet. For the purpose of planning this operation, bombs have *destruction distance* — the distance from the point of bomb's detonation to all the places where everything is considered destroyed.

Places for the bombs have been already selected. Now the Evil Empire wants to minimize the cost of production for the bombs. It is expensive to design atomic bombs tailored for different destruction distances, so the idea is to design a bomb with a specific destruction distance and to produce bombs according to this design for all the selected places. The problem is to find the minimal required destruction distance for the bomb design, so that destruction of the whole world is ensured.



For this problem the world is modeled as a sphere of a unit radius. The coordinates of the selected points for the bombs are specified in geographic coordinate system with latitude ϕ ($-90° < \phi < 90°$) and longitude λ ($-180° < \lambda \le 180°$). Latitude is the angle between a point and the equator, and longitude is the angle between a point and the prime meridian. The bombs are never placed on the poles, so their latitude is always less than 90 degrees by its absolute value.

Distances for destruction purposes are measured on the sphere. For example, the distance between the poles is exactly π. The world is considered destroyed if the distance from any point on the sphere to the closest bomb is less or equal to the destruction radius.

Input

The first line of the input file contains a single integer number n ($1 \le n \le 20$) — the number of the bombs. The following n lines describe the places of the bombs. Each line contains two integer numbers ϕ_i and λ_i ($-90 < \phi_i < 90$, $-180 < \lambda_i \le 180$) — latitude and longitude of the bomb. No two bombs are situated in the same place.

Output

Write to the output file a single number — the minimal destruction radius of the bombs that ensures destruction of the whole world. The answer must be precise up to 10^{-6}.

Sample input and output

knockdown.in	knockdown.out
4 59 30 53 83 41 69 41 41	2.864479