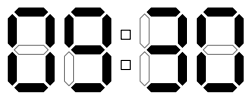
Problem A. Alarm Clock

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

Alice likes her digital alarm clock. She sets them up every evening. Last night Alice had a dream about her clock. Unfortunately, the only thing she is able to remember is the number of highlighted segments of the clock. Alice wonders what time was set on the clock in her dream.

Alice's clock have four digits: two for hours and two for minutes. For example, the clock below shows 9:30 (note the leading zero).



The clock uses following digit representation.

Input

The only line of the input file contains single integer n — the number of highlighted segments of the clock in Alice's dream ($0 \le n \le 30$).

Output

Output five characters in "`hh:mm`" format — the time shown on the clock in Alice's dream. The time must be correct: $0 \le$ `hh` < 24 and $0 \le$ `mm` < 60. If there are many possible correct times, output any of them. If there is none, output "`Impossible`".

Examples

alarm.in	alarm.out
23	09:30
28	Impossible
2	Impossible

Problem B. Buffcraft

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

Brenda enjoys a new role-playing game *Buffcraft*. Shields, swords, books and other carry-on items do not affects character stats in *Buffcraft*. The only way to increase the stats of your character is to buff her.

There are two types of buffs in *Buffcraft*. Direct buffs increase a base value of the stat, while percentage buffs increase stats by the fraction of the base value. To be precise, if unbuffed base value of your character stat is b, you have buffed her using n direct buffs of strength $d_1, d_2, ...d_n$ and m percentage buffs of strength $p_1, p_2, ..., p_m$, the resulting stat will be equal to $(b + d_1 + d_2 + \cdots + d_n)(100 + p_1 + p_2 + \cdots + p_m)/100$. Note that the resulting stat may be fractional.

Unfortunately, your character has only k buff slots and if you apply more than k buffs on her, only the last k buffs remains active. Thus, there is no reason to apply more than k buffs simultaneously. You cannot apply the same buff more than once.

Brenda is going to send his character to raid and wants to buff her health to maximal possible value. She has some direct and some percentage buffs at her disposal and needs your help to select the set of buffs that leads to maximal possible total health.

Input

The first line of the input file contains four integers b, k, c_d and c_p — the base health of the character, the number of buff slots, the number of available directs buffs, and the number of available percentage buffs.

The following line contains c_d integers d_i — strengths of direct buffs.

The last line of the input file contains c_p integer numbers p_i — strengths of percentage buffs.

All numbers in the input file are greater than or equal to zero, and less than or equal to fifty thousand.

Output

The first line of the output file must contain two integers n and m — the number of direct and percentage buffs to use ($0 \le n \le c_d$; $0 \le m \le c_p$; $0 \le n + m \le k$).

The following line must contain n different numbers — indices of direct buffs to apply (buffs are numbered from one).

The last line of the output must contain m different numbers — indices of percentage buffs to apply (also numbered from one).

The resulting total health after application of all $n + m$ buffs must be maximal possible.

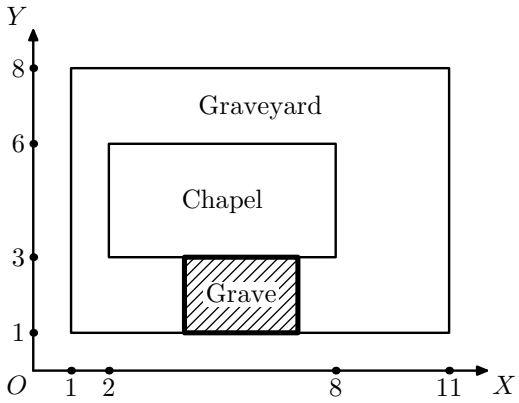
Examples

buffcraft.in	buffcraft.out
70 3 2 2 40 30 50 40	2 1 2 1 1
1 2 3 4 6 6 5 8 10 7 9	2 0 1 2

Problem C. Grave

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

Gerard develops a Halloween computer game. The game is played on a rectangular graveyard with a rectangular chapel in it. During the game, the player places new rectangular graves on the graveyard. The grave should completely fit inside graveyard territory and should not overlap with the chapel. The grave may touch borders of the graveyard or the chapel.



Gerard asked you to write a program that determines whether it is possible to place a new grave of given size or there is no enough space for it.

Input

The first line of the input file contains two pairs of integers: x_1, y_1, x_2, y_2 ($-10^9 \le x_1 < x_2 \le 10^9$; $-10^9 \le y_1 < y_2 \le 10^9$) — coordinates of bottom left and top right corners of the graveyard. The second line also contains two pairs of integers x_3, y_3, x_4, y_4 ($x_1 < x_3 < x_4 < x_2$; $y_1 < y_3 < y_4 < y_2$) — coordinates of bottom left and top right corners of the chapel.

The third line contains two integers w, h — width and height of the new grave ($1 \le w, h \le 10^9$). Side with length w should be placed along OX axis, side with length h — along OY axis.

Output

The only line of the output file should contain single word: "`Yes`", if it is possible to place the new grave, or "`No`", if there is not enough space for it.

Examples

grave.in	grave.out
1 1 11 8 2 3 8 6 3 2	Yes
1 1 11 8 2 3 8 6 4 3	No

Problem D. Arrangement of Contest

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

Little Dmitry and little Petr want to arrange a contest. Their little friends submitted several task proposals and now Dmitry and Petr want to select some of them for the contest. As they are just little boys, they cannot estimate quality of tasks, but they know for sure that in *good* contest title of the first problem starts with A, the title of the second one — with B, and so on.

Given titles of the proposed tasks, help little brothers to determine the maximal number of problems in a *good* contest they can arrange.

Input

The first line contains single integer n — the number of problem proposals received by the little brothers ($1 \le n \le 100$).

Next n lines contain titles of proposed problems, one per line. The length of each title does not exceed 30 characters. Each title starts with an uppercase letter and contains only English letters, digits and underscores.

Output

Output a single number — the maximal number of problems in a *good* contest. In case there is no *good* contest that may be arranged, output 0.

Examples

arrange.in	arrange.out
12 Arrangement_of_Contest Ballot_Analyzing_Device Correcting_Curiosity Dwarf_Tower Energy_Tycoon Flight_Boarding_Optimization Garage Heavy_Chain_Clusterization Intellectual_Property J Kids_in_a_Friendly_Class Lonely_Mountain	12
3 Snow_White_and_the_7_Dwarfs A_Problem Another_Problem	1
2 Good_Problem Better_Problem	0

Problem E. Ballot Analyzing Device

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

Election committee of Flatland is preparing for presidential elections. To minimize human factor in ballot counting they decided to develop an automated Ballot Analyzing Device (BAD).

There are n candidates running for president. The ballot contains one square field for each candidate. The voter must mark exactly one of the fields. If no field is marked or there are two or more marked fields, the ballot is invalid. Each voter puts his/her ballot to a special scanner in BAD. The scanner analyzes marks on the ballot and creates a special *voting string* of n characters: '`X`' for marked field and '`.`' for unmarked one. Now voting strings must be analyzed to get the report. Your task is to develop a report generator for BAD.

Given voting strings for all ballots, your program must print the voting report. Candidates in the report must be arranged in order of decreasing number of votes. If two candidates have the same number of votes, they must have the same order as in a voting ballot. For each candidate calculate his/her result in percent (if the candidate received p votes, the result in percent is $100p/m$). The last line of the report must indicate the percentage of the invalid ballots.

Input

The first line contains two integers n and m — the number of candidates and the number of ballots ($2 \le n \le 10$; $1 \le m \le 1000$). The following n lines contain last names of candidates. Each name is a string of at most 100 English letters. There is no candidate named "`Invalid`".

Then m lines follow, each of them contains one voting string.

Output

Print $n+1$ lines. First print results for candidates in percent. For each candidate print his/her last name followed by a space and then his/her result in percent and a percent sign '`%`'. The last line must specify the percentage of invalid ballots: a word "`Invalid`" followed by a space, the percentage of invalid ballots and a percent sign.

Round all numbers to exactly two digits after the decimal point. If the number is exactly in the middle of two representable numbers, output the greater one (e.g. output "`12.35`" for 12.345).

Example

bad.in	bad.out
4 7 Loudy Apples Dogman Miller .X.. X... .... ..X. ..XX ..X. ..X.	Dogman 42.86% Loudy 14.29% Apples 14.29% Miller 0.00% Invalid 28.57%

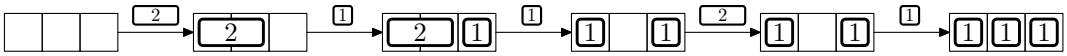
Problem F. Energy Tycoon

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

Little Vasya is playing a new computer game — turn-based strategy "Energy Tycoon".

The rules of the game are quite simple:

- The board contains n slots arranged in a line.
- There are power plants, one power plant occupies one or two consecutive slots, and produces one unit of energy.
- Each turn the game allows you to build one new power plant, you can put it on the board if you wish. If there is no place for the new power plant, you can remove some older power plants.
- After each turn, the computer counts the amount of energy produced by the power plants on the board and adds it to the total score.



Vasya already knows the types of power plant he will be able to build each turn. Now he wants to know, what the maximum possible score he can get is. Can you help him?

Input

The first line of the input contains one integer n ($1 \le n \le 100\,000$) — the number of slots on the board. The second line contains the string s. The i-th character of the string is 1 if you can build one-slot power plant at the i-th turn and the character is 2 if you can build two-slot power plant at the i-th turn. The number of turns does not exceed 100 000.

Output

The output should contain a single integer — the maximal score that can be achieved.

Examples

energy.in	energy.out
3 21121	10
2 12	2
2 211	4

The picture shows the optimal sequence of moves for the first example.

Problem G. Garage

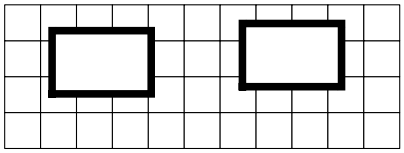
Input file:	`garage.in`
Output file:	`garage.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Wow! What a lucky day! Your company has just won a social contract for building a garage complex. Almost all formalities are done: contract payment is already transferred to your account.

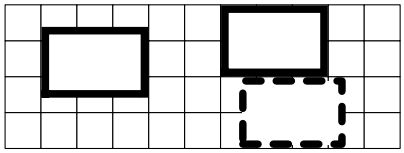
So now it is the right time to read the contract. Okay, there is a sandlot in the form of $W \times H$ rectangle and you have to place some garages there. Garages are $w \times h$ rectangles and their edges must be parallel to the corresponding edges of the sandlot (you may not rotate garages, even by 90°). The coordinates of garages may be non-integer.

You know that the economy must be economical, so you decided to place as *few* garages as possible. Unfortunately, there is an opposite requirement in the contract: placing maximum possible number of garages.

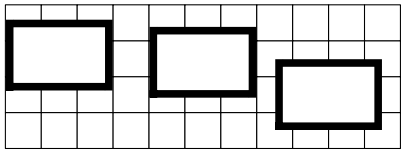
Now let's see how these requirements are checked... The plan is accepted if it is impossible to add a new garage without moving the other garages (the new garage must also have edges parallel to corresponding sandlot edges).



Accepted optimal plan



Rejected plan



Accepted, but non-optimal plan

Time is money, find the minimal number of garages that must be ordered, so that you can place them on the sandlot and there is no place for an extra garage.

Input

The only line contains four integers: W, H, w, h — dimensions of sandlot and garage in meters. You may assume that $1 \le w \le W \le 30\,000$ and $1 \le h \le H \le 30\,000$.

Output

Output the optimal number of garages.

Examples

garage.in	garage.out
11 4 3 2	2
10 8 3 4	2
15 7 4 2	4

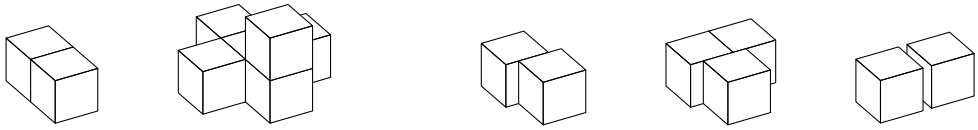
The plan on the first picture is accepted and optimal for the first example. Note that a rotated (2×3) garage could be placed on the sandlot, but it is prohibited by the contract.

Problem H. Aztec Pyramid

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

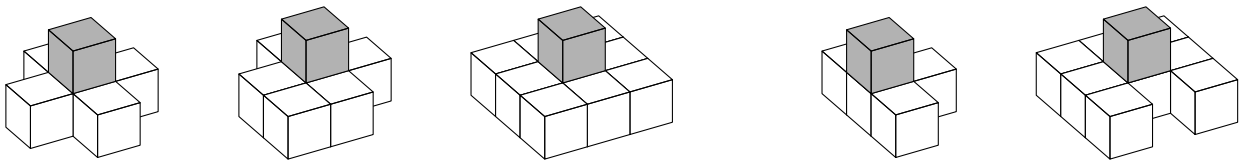
Aztec emperor Cuitláhuac is going to build a pyramid in his honor. This pyramid should be taller than previous ones.

The Aztec pyramid is build out of stone blocks. Each block is 1×1×1-hunab cube. Cuitláhuac places first block on the ground during the foundation ceremony. Each of the following blocks must share a face with at least one of the previous blocks.



Valid block placements Invalid block placements

The block is stable if it stands on the ground, or it stands on another block, that has a block or the ground next to each face. To stand the test of time the pyramid must be stable i.e. each block of it must be stable.



Stable blocks Unstable blocks

Cuitláhuac asks you to determine the height of the tallest stable pyramid that can be built out of available blocks.

Input

The only line of the input file contains a single integer number n — the number of available blocks, including the first one ($1 \le n \le 10^9$).

Output

Output the height of the tallest stable pyramid that may be built out of n blocks. The height must be output in hunabs.

Examples

aztec.in	aztec.out
6	2
5	1
20	3

Problem I. Battleship

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

You are in a computer security business. Recently one of your clients developed a new test designed to separate humans from programs. Your task is to test its efficiency.

The test is based on a variation of the Battleship game. This game is played on a ten by ten grid. Before the beginning of the test, you should place ten ships on this grid. Each ship is represented by a number of consecutive cells, arranged vertically or horizontally. You should place exactly one four-cell ship, two three-cell ships, three two-cell and four one-cell ships. No two ships should have any common or adjacent cells. They should not share a corner of a cell as well.

After the ships have been arranged, the test proceeds in a number of rounds. At each round one cell is announced to be "shot". If this cell is occupied by a ship, this ship is considered to be "hit". After each cell of a ship has been hit at least once, the ship sinks. The test ends after all the ships have sunk.

Let's call the *complexity* of an arrangement the number of round it took to finish the tests. The idea is that humans would create much more complex arrangements than programs.

You have already figured out that the cells are shot in a predefined order, each cell being shot exactly once. Now you want to write a program that would create the most complex arrangement on the basis of the order in which the cells are shot.

Input

The input file consists of ten lines, containing ten numbers each. Each number represents the round at which the corresponding cell will be shot. All numbers are distinct and belong to the range $1 \ldots 100$.

Output

Output the optimal ship arrangement for the battleship test. Empty cells should be represented by the dot ('`.`'), occupied cells should by represented by the number sign ('`#`').

Examples

battleship.in	battleship.out
1 2 3 4 5 6 7 8 9 10 36 37 38 39 40 41 42 43 44 11 35 64 65 66 67 68 69 70 45 12 34 63 84 85 86 87 88 71 46 13 33 62 83 96 97 98 89 72 47 14 32 61 82 95 100 99 90 73 48 15 31 60 81 94 93 92 91 74 49 16 30 59 80 79 78 77 76 75 50 17 29 58 57 56 55 54 53 52 51 18 28 27 26 25 24 23 22 21 20 19	...####... .... #....##... #.#.....#. ....#. ...###.... .#........ ....#. ..#.....#. .....#..#.

Problem J. Deepest Station

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

> The Saint Petersburg Metro is the underground railway system in Saint Petersburg and Leningrad Oblast, Russia. It has been open since November 15, 1955. Formerly known as the V.I. Lenin Order of Lenin Leningrad Metropoliten, the system exhibits many typical Soviet designs and features exquisite decorations and artwork making it one of the most attractive and elegant metros in the world. Due to the city's unique geology, the Saint Petersburg Metro is one of the deepest subway systems in the world and the deepest by the average depth of all the stations. The system's deepest station, Admiralteyskaya, is 105 metres below the ground. Serving two and a half million passengers daily, it is also the 12th busiest subway system in the world.
>
> From Wikipedia, the free encyclopedia

After building *Admiralteyskaya* metro station, the government of *Saint Petersburg* decided to build the really deep station which would be the deepest in the world. It will be d meters under the ground! The station will be built right under the *Smolny Sobor* and for use by officials only. The Department of Urban Development has its internal coordinate system for building a project. The origin of this system point is exactly the Smolny Sobor and applicate means depth under the ground. So, the new station will have coordinates $(0, 0, d)$.

Due to security reasons station's lobby must be located outside the *Smolny Convent*, at point (x, y). Your task is to help the government with building moving staircases. The station will use innovative staircases that should go down at angle 45°. It is possible to build one intermediate lobby somewhere underground (like at Admiralteyskaya) and two staircases. Given x, y and d, find coordinates of intermediate lobby. Note that you cannot dig below d meters under the ground, so the intermediate lobby must not be deeper than the main station.

Input

The only line of the input file contains three integer numbers: x, y and d — the coordinates of the station in the Department of Urban Development coordinate system ($-10\,000 \le x, y \le 10\,000$; $(x, y) \ne (0, 0)$; $106 \le d \le 10\,000$).

Output

Output three numbers with precision at least 10 digits after decimal point: coordinates of the intermediate lobby in the Department of Urban Development coordinate system.

If it is impossible to build staircases, output "`Impossible`". If no intermediate lobby is required, output "`Single staircase`".

Examples

deepest.in	deepest.out
0 100 300	0.0 200.0 100.0
300 400 500	Single staircase
400 400 500	Impossible

Problem K. Electricity

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

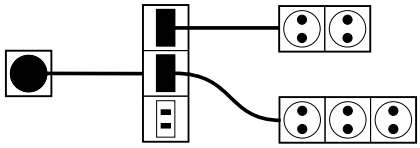
You are managing a power network for a programming competition and you have to connect a lot of computers to the power supply. Unfortunately, there are two standards for electrical plug and socket: A and B. These standards are incompatible, so the plug of standard A can only be plugged in the socket of standard A and the plug of standard B can only be plugged in the socket of standard B.

In the main competition hall, there is exactly one main socket of type A. Every computer that will be used in this programming competition has one plug of type A. Thereby only one computer can be connected directly to the main socket. But you have a number of power strips of two types.

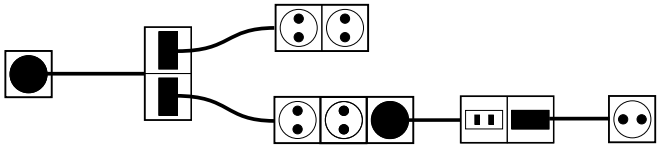
- Power strips of the first type have one plug of type A and several sockets of type B.
- Power strips of the second type have one plug of type B and several sockets of type A.

All the power strips are very powerful and can withstand any load. So you can create a power network by connecting one power strip of the first type to the main socket, then some power strips of the second type to this power strip of the first type, etc. At the end you will get several available sockets of type A for computers.

Your task is to find the maximum number of computers that can be connected to the power network, using available power strips.



Possible solution for the first example.



Possible solution for the second example.

Input

The first line of input file contains two integer numbers n and m — the number of power strips of the first and the second type ($0 \le n, m \le 100\,000$).

The second line contains n integer numbers a_i — the number of sockets on the power strips of the first type ($1 \le a_i \le 1000$).

The second line contains m integer numbers b_i — the number of sockets on the power strips of the second type ($1 \le b_i \le 1000$).

Output

Output the maximum number of computers that can be connected to the power network.

Examples

electricity.in	electricity.out
3 2 3 2 1 2 3	5
2 3 2 2 2 3 1	5

Problem L. Final Standings

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

Formula TheByte is the most famous race competition in *ByteLand*. The competition is over and each of n drivers has a number of points. A driver that has more points is placed higher.

Final standings are not disclosed yet, but we know that the total number of points earned by all the drivers is p and there are only d different numbers of points among top k drivers.

The *ByteLand Times* asks you to guess final standings based on the given information.

Input

The only line of the input file contains four integer numbers: n, p, k and d — the number of drivers, the number of points, the number of top drivers, and the number of different numbers of points among top k drivers ($1 \le k \le n \le 1000$; $0 \le p \le 1\,000\,000$; $1 \le d \le k$).

Output

Output the possible standings that match given n, p, k and d.

If it is possible to create the correct final standings, you should output final standings. The i-th line of the final standings must contain a number of points earned by i-th driver. Drivers should be ordered by number of points in the descending order.

If there is no possible final standings satisfying the given data, output the single line "`Wrong information`".

Examples

final.in	final.out
3 4 2 2	2 1 1
3 5 2 2	3 2 0
2 5 2 1	Wrong information