Problem A. Overdraft

Source file name:	Overdraft.c, Overdraft.cpp, Overdraft.java, Overdraft.py
Input:	`Standard`
Output:	`Standard`

Banks often charge overdraft fees if you attempt to withdraw more money from your account than is available in your current balance. Given a sequence of deposits and withdrawals (and assuming each deposit and withdrawal is immediately reflected in your balance), determine the minimum (non-negative) starting balance you need to ensure that you will not be charged any overdraft fees over the course of the sequence.

Input

The first line of input contains a single integer n $(1 \leq n \leq 1000)$, which is the number of transactions.

Each of the next n lines contains a single integer t $(-10^6 \leq t \leq 10^6,\ t \neq 0)$. These are the transactions, in the order that they occur. A positive number represents a deposit, a negative number represents a withdrawal. No two transactions occur simultaneously.

Output

Output a single non-negative integer, which is the minimum non-negative balance you must start with in your account in order to avoid any overdraft fees.

Example

Input	Output
3 3 -5 3	2

Problem B. Black and White

Source file name:	Black.c, Black.cpp, Black.java, Black.py
Input:	`Standard`
Output:	`Standard`

Black and White is a Chinese children's game played in rounds. During each round, the children who are playing all put their hands in either face-up ("White") or face-down ("Black"). If all the children but one make the same choice, then the "odd one out" sits out for the rest of the game. Play continues until there are only two children left.

Each child independently chooses whether to put their hand face-up with their own fixed probability. What is the expected number of rounds that such a game will last?

Input

The first line contains a single integer n ($2 \leq n \leq 20$), which is the number of children.

Each of the next n lines contains a single real number p ($0.1 \leq p \leq 0.9$). These are the probabilities for each child that they will put their hand in face-up. The probabilities will have at most three digits after the decimal point.

Output

Output a single real number, which is the expected number of rounds. The result must be accurate to within an absolute error of 10^{-6}.

Example

Input	Output
3 0.5 0.5 0.5	1.3333333
3 0.3 0.3 0.3	1.5873015
5 0.1 0.3 0.5 0.7 0.9	7.4752846

Problem C. Double Password

Source file name:	Password.c, Password.cpp, Password.java, Password.py
Input:	`Standard`
Output:	`Standard`

A computer at ICPC headquarters is protected by a four-digit password–in order to log in, you normally need to guess the four digits exactly. However, the programmer who implemented the password check left a backdoor in the computer–there is a second four-digit password. If the programmer enters a four-digit sequence, and for each digit position the digit entered matches at least one of the two passwords in that same position, then that four-digit sequence will log the programmer into the computer.

Given the two passwords, count the number of distinct four-digit sequences that can be entered to log into the computer.

Input

The input consists of exactly two lines. Each of the two lines contains a string s ($|s| = 4,\ s \in \{0-9\}^*$). These are the two passwords.

Output

Output a single integer, which is the number of distinct four-digit sequences that will log the programmer into the system.

Example

Input	Output
1111 1234	8
2718 2718	1

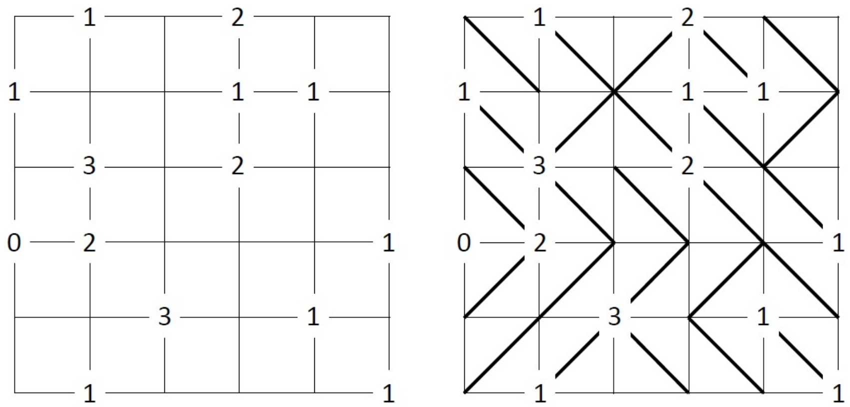
Problem D. Diagonals

Source file name: Diagonals.c, Diagonals.cpp, Diagonals.java, Diagonals.py
Input: `Standard`
Output: `Standard`

Diagonals is a pencil puzzle which is played on a square grid. The player must draw a diagonal line corner to corner in every cell in the grid, either top left to bottom right, or bottom left to top right. There are two constraints:

- Some intersections of gridlines have a number from 0 to 4 inclusive on them, which is the exact number of diagonals that must touch that point.
- No set of diagonals may form a loop of any size or shape.

The following is a 5×5 example, with its unique solution:



Given the numbers at the intersections of a grid, solve the puzzle.

Input

The first line of input contains an integer n ($1 \le n \le 8$), which is the size of the grid.

Each of the next $n+1$ lines contains a string s ($|s| = n+1,\ s \in \{0,\ 1,\ 2,\ 3,\ 4,\ +\}^*$). These are the intersections of the grid, with '+' indicating that there is no number at that intersection.

The input data will be such that the puzzle has exactly one solution.

Output

Output exactly n lines, each with exactly n characters, representing the solution to the puzzle. Each character must be either '/' or '\'.

Note that Example 1 corresponds to the example in the problem description.

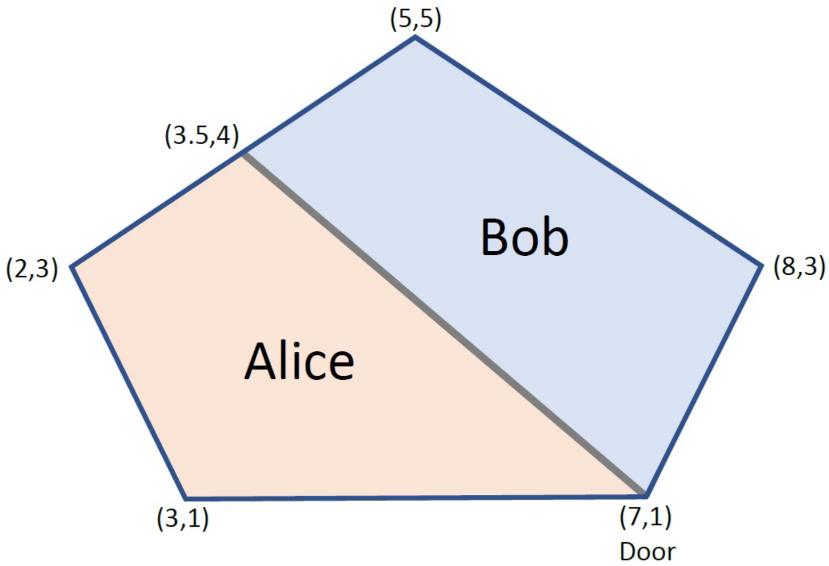
Example

Input	Output
5 +1+2++ 1++11+ +3+2++ 02+++1 ++3+1+ +1+++1	\\/\\ \/\\/ \\\\\ ////\ //\\\
3 ++++ +1+1 +31+ +0+0	/\/ /// /\/
4 +++++ +3++2 ++3++ +3+3+ ++2+0	\//\ \\// \\\/ /\//

Problem E. Dorm Room Divide

Source file name:	Dorm.c, Dorm.cpp, Dorm.java, Dorm.py
Input:	`Standard`
Output:	`Standard`

Bob and Alice are roommates at the International College of Polygonal Chambers (ICPC). To avoid conflict, they've agreed to divide their dorm room in half–as closely as possible. However, the room is shaped so irregularly that they need your help!



Each dorm room is a convex polygon, with a single entrance. You need to figure out how to divide this room in half (by area) using a single straight line starting at the door, and terminating on a wall or corner of the room.

Input

The first line of input contains a single integer n ($3 \le n \le 2 \cdot 10^5$), which is the number of vertices describing the convex polygon.

Each of the next n lines contains two space-separated integers x and y ($-10^7 \le x,\ y \le 10^7$). These are the coordinates of the vertices of the convex polygon, in counterclockwise order. All points will be distinct.

The door is considered to be a single point located at the first vertex given in the input.

Output

Output two space-separated real numbers, which are the x and y coordinates of the other endpoint of the dividing line, such that the area of the room is divided in half. Each coordinate value must be accurate to within an absolute error of 10^{-6}. Output x first, then y.

Note that Example 1 corresponds to the example in the problem description.

Example

Input	Output
5 7 1 8 3 5 5 2 3 3 1	3.5 4
3 2 2 10 3 6 8	8 5.5

Problem F. Rise and Fall

Source file name:	Rise.c, Rise.cpp, Rise.java, Rise.py
Input:	`Standard`
Output:	`Standard`

A number is said to *Rise and Fall* if the decimal representation can be broken up into two parts (possibly empty) where the first part has digits in nondecreasing order and the second part has digits in nonincreasing order.

Compute the largest number less than or equal to an input number that rises and falls.

Input

The first line of input contains an integer t $(1 \le t \le 10^5)$, which is the number of test cases.

Each of the next t lines contains a single integer n $(1 \le n < 10^{100000})$. Each is a single test case.

Note: that is not a typo. The integer can be up to 10^5 digits long.

The sum of the lengths of all input test cases will not exceed 10^5.

Output

For each test case, output a single line with a single integer, which is the largest number less than or equal to the n for that test case that rises and falls.

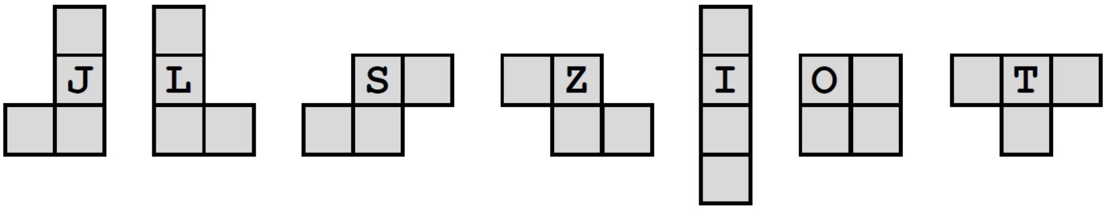
Example

Input	Output
2 29041 56577	29000 56555

Problem G. Tetris Generation

Source file name:	Tetris.c, Tetris.cpp, Tetris.java, Tetris.py
Input:	`Standard`
Output:	`Standard`

The classic game Tetris involves arranging falling tetrominoes on a board. There are seven different tetrominoes, each named after a letter that resembles their shape: `J`, `L`, `S`, `Z`, `I`, `O`, and `T`.



In the original Tetris, the player would receive one tetromino at a time, and each tetromino would be chosen from among the seven possibilities independently and uniformly at random. This meant that any sequence of tetrominoes could appear in a game, such as numerous `I` tetrominoes in a row. Modern versions of Tetris remove these streaks by generating tetrominoes in groups of seven: The first seven tetrominoes in a game will be one of each of the seven different tetrominoes in a random order. The next seven tetrominoes will also be one of each of the seven different tetrominoes in a random order (possibly but not necessarily different from the ordering of the first seven). Same goes for the next seven, and so on and so forth. With this generator, it is still possible to get two of the same tetromino in a row (for example, the seventh and eighth tetrominoes in the game can be the same as each other), but it is not possible to get three of the same type in a row.

Given a sequence of tetrominoes, determine whether it is possible for a modern Tetris generator to produce that sequence at some point in a game.

Input

The first line of input contains an integer t ($1 \le t \le 10^5$), which is the number of test cases.

Each of the next t lines contains a single string s ($1 \le |s| \le 1000$, $s \in \{\mathrm{J, L, S, Z, I, O, T}\}^*$). This string represents a sequence of tetrominoes, and is a single test case.

The sum of the lengths of all input test cases will not exceed 10^5.

Output

For each test case, output a single line with a single integer, which is 1 if the sequence can be generated by a modern Tetris generator, and 0 otherwise.

Example

Input	Output
2 JJTO JJTT	1 0

Problem H. Hopscotch 500

Source file name:	Hopscotch.c, Hopscotch.cpp, Hopscotch.java, Hopscotch.py
Input:	`Standard`
Output:	`Standard`

Do you remember the new art installation from NAC 2020? Well, that artist is at it again, on a grander scale this time, and the new artwork still inspires you–to play a childish game. The art installation consists of a floor with a square matrix of tiles. Each tile holds a single number from 1 to k.

You want to play hopscotch on it! You want to start on some tile numbered 1, then hop to a tile numbered 2, then 3, and so on, until you reach a tile numbered k.

Instead of the usual Euclidean distance, define the distance between the tile at $(x_1,\ y_1)$ and the tile at $(x_2,\ y_2)$ as:

$$\min\left[(x_1 - x_2)^2,\ (y_1 - y_2)^2\right]$$

You want to hop the shortest total distance overall, using this new distance metric. Note that a path with no hops is still a path, and has length 0. What is the length of the shortest path?

Input

The first line of input contains two space-separated integers n ($1 \le n \le 500$) and k ($1 \le k \le n^2$), where the art installation consists of an $n \times n$ matrix with tiles having numbers from 1 to k.

Each of the next n lines contains n space-separated integers x ($1 \le x \le k$). These are the numbers in the art installation.

Output

Output a single integer, which is the total length of the shortest path from any 1 tile to any k tile using our distance metric, or -1 if no such path exists.

Example

Input	Output
10 5 5 1 3 4 2 4 2 1 2 1 4 5 3 4 1 5 3 1 1 4 4 2 4 1 5 4 5 2 4 1 5 2 1 5 5 3 5 2 3 2 5 5 2 3 2 3 1 5 5 5 3 4 2 4 2 2 4 4 2 3 1 5 1 1 2 5 4 1 5 3 2 2 4 1 2 5 1 4 3 5 5 3 2 1 4 3 5 2 3 1 3 4 2 5 2 5 3 4 4 2	0
10 30 18 13 30 15 18 16 14 1 5 5 17 18 7 30 14 30 13 14 1 28 28 24 7 23 9 10 5 12 21 6 11 16 6 2 27 14 1 26 7 21 16 2 9 26 6 24 22 12 8 16 17 28 29 19 4 6 21 19 6 22 11 27 11 26 13 23 10 3 18 6 14 19 9 8 17 6 16 22 24 1 12 19 10 21 1 8 20 24 29 21 21 29 1 23 23 24 6 20 25 17	19

Problem I. XOR Island

Source file name:	Island.c, Island.cpp, Island.java, Island.py
Input:	`Standard`
Output:	`Standard`

On an island populated entirely with perfect logicians, each islander is wearing a hat that displays a positive integer. Each islander can see all other islanders' hats, but they cannot see their own hat. An islander has no information about the number on their own hat, other than the fact that it is a positive integer.

One day, a mysterious message appears in the sky and says "There exist three distinct islanders such that the XOR of the integers on two of their hats is the integer on the hat of the third". After this message appears, the islanders schedule meetings for several days in a row, one meeting per day. At each meeting any islander who knows for sure that they are part of some triple that satisfies the message will raise their hand.

Assuming that no islanders lie or make mistakes, and each will raise their hand as soon as it is possible for them to know that they are part of a triple, how many days will it take for at least one islander to raise their hand at a meeting?

Input

The first line of input contains a single integer n $(3 \leq n \leq 25)$, the number of islanders.

Each of the next n lines contains a single integer a $(1 \leq a < 2^{25})$, which are the positive integers on the islanders' hats.

It is guaranteed that the input is chosen such that the message in the sky is true; there is at least one triple among the islanders' hats such that one hat's integer is equal to the XOR of the integers on the other two hats.

Output

Output a single integer, which is the number of daily meetings it will take for some islander to figure out that their hat is part of some XOR triple. It can be proven that at least one person will eventually raise their hand.

Example

Input	Output
3 1 2 3	1
11 9 1 14 2 11 7 6 7 6 5 3	3

Problem J. Who Goes There?

Source file name:	Whogoes.c, Whogoes.cpp, Whogoes.java, Whogoes.py
Input:	Standard
Output:	Standard

What happens when more teams want to go to an ICPC regional site than the site has capacity for? Who goes there?

One possible policy is the following: Every school is allowed to register as many teams as they wish. Accept every school's first team, then accept every school's second team (for schools with more than one team), then third, and so on, until all teams are accepted, or there isn't enough capacity for the next wave. Then, if there are extra spots available, the spots are given to schools, one by one, in the order that the schools registered.

Given the capacity of a site, the number of teams registered by each school and the order that they registered, determine how many teams from each school are accepted.

Input

The first line of input contains two integers n ($1 \leq n \leq 100$) and m ($1 \leq m \leq 100$), where n is the capacity of the site and m is the number of schools that wish to compete there.

Each of the next m lines contains an integer t ($1 \leq t \leq 100$), which is the number of teams that a school has registered. The schools are listed in the order that they registered.

Output

Output m lines, one for each school. Each line must contain a single integer indicating the number of teams accepted from that school. Output them in the same order as they appear in the input.

Example

Input	Output
20 5 7 5 1 6 12	5 5 1 5 4