Problem A: Rayan

The World Finals of the **Rayan International Programming Contest** were held in November 2025 in Tehran, Iran, at Sharif University of Technology. Top programmers from 25 countries gathered to compete in this international event.

Now that the contest has concluded, the organizers are preparing certificates for all participants. Each certificate must display a title based on the contestant's final rank, according to the following rules:

- Contestants with ranks 1 to 10 receive the title **"Ranked i"**, where i is the contestant's rank number.
- Contestants with ranks 11 to 20 receive the title **"Highest Honors"**.
- Contestants with ranks 21 to 30 receive the title **"High Honors"**.
- All other contestants receive the title **"Honorable Mention"**.

Mahya, from the Rayan office, is responsible for preparing the certificates. She needs your help to determine the correct title for each contestant based on their rank.

Write a program that determines the appropriate certificate title given a contestant's rank.

Input

A single integer n ($1 \leq n \leq 100$) — the rank of the contestant.

Output

A single line containing the title that should appear on the contestant's certificate.

Example

Standard Input	Standard Output
`4`	`Ranked 4`
`15`	`Highest Honors`
`27`	`High Honors`
`45`	`Honorable Mention`

Problem B: MicroMasters

The Computer Engineering Department of Sharif University offers a series of MicroMasters courses designed to enhance students' skills in cutting-edge technologies. Each course provides a valuable certificate from Sharif University, which can significantly boost your academic and professional profile.

Ali wants to enroll in one of these MicroMasters courses for the first time. Each course has a price, and a special first-time registration discount in percent, which may vary from course to course.

Ali is a savvy student who wants to obtain the Sharif certificate while paying the minimum possible amount. Your task is to help Ali choose the course that will cost him the least, taking into account the first-time discount for each course.

Input

The first line contains a single integer n ($1 \leq n \leq 10^3$), number of available courses.

The i^{th} line of the next n lines contains two integers c_i and p_i ($1 \leq c_i \leq 10^6$; $0 \leq p_i \leq 100$), the original price of the course i and the first-time registration discount in percent for this course.

Output

Print a single integer, the number of the course Ali should register for to pay the minimum amount. If multiple courses have the same minimum cost, print the smallest course number among them.

Example

Standard Input	Standard Output
5 100 20 150 50 200 0 120 30 180 60	5

Problem C: Genome Similarity Analyzer

In a bioinformatics research lab, scientists study an $n \times n$ matrix S representing similarity between beaver and duck genomes. $S_{i,j}$ stores a non-negative integer indicating the similarity between marker i of beaver genome and marker j of duck genome.

A sequence $(x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k)$ of arbitrary length k where $1 \le x_i, y_i \le n$, is called a *matching sequence*. A matching sequence is considered *refined* if for every i and j $(i < j)$, we have $x_i < x_j$ and $y_i < y_j$.

The *similarity score* of a refined matching sequence is defined as the sum of $S_{x_1,y_1}, S_{x_2,y_2}, \cdots, S_{x_k,y_k}$. The *score* of a matrix is defined as the maximum similarity score of all possible refined matching sequences.

The laboratory uses an experimental $n \times (m + n - 1)$ matrix T, where for every i $(1 \le i \le n - 1)$, column $m + i$ is a copy of column i.

From this matrix, researchers extract m matrices $S_1, S_2, \cdots, S_m$, where S_i is an $n \times n$ matrix that contains columns i through $i + n - 1$ of matrix T.

For each matrix S_i, they want to compute its score.

Input

The first line of input contains two integers n and m $(1 \le n \le m \le 3000)$.

Each of the next n lines contains m integers, describing the first m columns of matrix T.

for every valid i and j, $0 \le T_{i,j} \le 100\,000$.

For each i and j that $m < j \le m + n - 1$, $T_{i,j}$ equals $T_{i,j-m}$.

Output

Print m space separated integers, the score of S_1 through S_m.

Example

Standard Input	Standard Output
3 3 2 0 3 0 1 0 1 0 2	5 4 4

Problem D: Sequence Extraction Machine

In an old research laboratory, Alice discovers a strange mechanical device labeled **Sequence Extraction Machine**. The machine accepts a sequence of integers and allows Alice to perform the following operation:

1. Alice chooses any two adjacent elements in the current sequence, say x and y.
2. The machine awards her exactly $|x - y|$ points.
3. Then it removes both elements from the sequence.
4. The remaining elements close the gap and form a shorter sequence.

This continues until the sequence becomes empty or just contains one item.

However, the order in which Alice removes adjacent pairs completely affects the total score. To maximize her reward, she must choose the optimal sequence of removals.

Your task is to help her compute the maximum total score she can obtain.

Input

The first line contains an integer n ($1 \leq n \leq 10^6$), the length of the sequence.

The second line contains n integers $a_1, \ldots, a_n$ ($0 \leq a_i \leq 10^9$).

Output

Print a single integer, the maximum possible total score.

Example

Standard Input	Standard Output
5 5 4 2 2 3	5
10 2 8 5 10 4 7 5 0 9 8	26

Problem E: Mom's Soup

A nasty flu is sweeping the city! Your only hope is your mother's "Miracle Soup", but as a competitive programmer, you realize that cooking is actually a complex optimization problem.

You are cooking the soup in your favorite cubic pot of side length N. To simplify the thermal dynamics, you model the soup as N^3 unit cells in an $N \times N \times N$ grid. Each cell contains soup at a single, uniform temperature. Initially, all cells are at temperature $0°$.

The four side walls and the bottom floor of the pot are heat sources. The top of the pot is open. The soup boils at temperature $L°$, so no cell can exceeds this temperature.

Initially, all cells are inactive. Then, the heating process occurs in discrete steps as follows:

- Activation Phase: At the start of each second, any inactive cell becomes active immediately if:
  - It is adjacent to the pot's walls or floor, OR
  - It has a neighbor (sharing a face) that is at least r degrees warmer than itself.
- Heating Phase: During the same second, all active cells increase their temperature by $1°$ (up to a maximum of $L°$).
- Persistence: Once a cell becomes active, it remains active until it reaches L, even if the neighbor condition is no longer met.

You can stir the soup any time, which takes t seconds. During stirring, no heating occurs. After stirring, every cell's temperature becomes the soup's average, rounded down:

$$\text{NewTemp} = \lfloor \frac{\sum \text{Temp}_{\text{all cells}}}{N^3} \rfloor$$

After stirring, all internal cells become inactive. However, cells adjacent to the walls or floor immediately become active again.

Your goal is to find the minimum time required to heat up the soup and make it ready to eat. The soup is ready when every cell reaches at least temperature k.

Input

The only line contains five integers N, L, r, t, k ($1 \le N, L, t \le 5000, 1 \le r, k \le L$).

It can be proven that a solution is always possible.

Output

Print one integer, the minimum time in seconds.

Example

Standard Input	Standard Output
`100 10 8 400 3`	`395`
`10 20 10 2 1`	`5`

Note

Sample 1: In this case, the optimal solution is to not stir at all.

- **Without Stirring:** After 395 seconds, temperature of all cells will be at least 3°.
- **With Stirring:** One possible plan with stirring is to wait 55 seconds for the soup's average temperature to reach 3°, then stir, resulting in $55 + 400 = 455$ seconds.
  Note that any plan with stirring takes at least $t = 400$ seconds.

Sample 2: The optimal solution is to wait for 3 seconds, then stir (taking 2 more seconds). Total time will be $3 + 2 = 5$ seconds. The entire soup is now at temperature 1°, which is at least $k = 1$.

Problem F: Symmetric Blister Pack

A pharmaceutical company produces medicine in rectangular blister packs, where each pill occupies one cell in an $R \times C$ grid. Initially, every position in the blister pack contains a pill. Over time, some pills have been taken, leaving empty positions. The company considers a blister pack to be well-organized if the **remaining pills** admit at least one axis of symmetry.

The axis of symmetry can be:

- Vertical: symmetry with respect to the central vertical axis.
- Horizontal: symmetry with respect to the central horizontal axis.

Note that for both types, the axis may pass either through the centers of cells (if the dimension is odd) or between two adjacent rows/columns (if the dimension is even).

You may remove additional pills from the pack. Your task is to determine the minimum number of additional pills that must be taken so that the remaining pills admit at least one axis of symmetry.

Input

The first line contains two integers R and C ($1 \leq R, C \leq 100$), the number of rows and columns.

The next R lines each contain a string of length C consisting only of:

- B: A pill is present.
- W: The pill has already been taken.

Output

Print one integer, the minimum number of additional pills that must be taken so that the remaining pills admit at least one vertical or horizontal axis of symmetry.

Example

Standard Input	Standard Output

`2 5`
`WBBBW`
`BWWBB`
	`1`

`3 5`
`WBBBB`
`BBWBB`
`BBBWB`
	`2`

Problem G: The Magical Beads

An archaeologist discovers an ancient necklace of beads containing magical powers. The magical power of each bead is represented by an integer value, which may be positive, zero, or negative. The power of the necklace equals the sum of the magical powers of all its beads.

To maximize the power of the necklace, the archaeologist may perform the following ritual any number of times (possibly zero):

- Choose three consecutive beads.
- Destroy the first and third beads, leaving the middle bead untouched.
- Join the remaining parts of the necklace, preserving their original order.

Naturally, when the necklace contains fewer than three beads, the ritual can no longer be performed.

Your task is to maximize the power of the final necklace.

Input

The first line contains an integer n ($1 \le n \le 3 \times 10^5$), the number of beads.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$), where a_i is the magical power of bead i.

Output

Print one integer: the maximum possible power of the necklace after performing the ritual any number of times.

Example

Standard Input	Standard Output
5 10 -4 -5 6 2	12

Problem H: Divar Knots

Divar has recently purchased a new building to serve as its new headquarters. The building is called "Knots" due to its knot-like forms featured in its exterior design. Divar is committed to a modern, ergonomic, and thoughtfully organized workplace. Accordingly, a senior architect has been brought in to design the internal structure and floor-by-floor layout.

Each office floor is an $n \times m$ rectangular grid of rooms, completely surrounded by exterior walls. Each room can have up to 4 adjacent rooms in the 4 cardinal directions: North, West, South, and East. Initially, every pair of adjacent rooms is separated by a partition wall. However, a door may be installed in a partition wall to connect the two adjacent rooms.

Hessam wants to install a set of doors – called a *door layout* – such that:

- All rooms become connected through doors.
- There is exactly one path between any two rooms.

In addition, the door layout must satisfy the following conditions:

- No room may have doors connecting it to both its North-adjacent room and its West-adjacent room.
- Exactly k rooms must be *meeting rooms*. A room will be a meeting room, if and only if it has **exactly one** door.

Your task is to compute the number of distinct door layouts satisfying all of the above conditions. Two door layouts are considered different if and only if there exists at least one partition wall where it has a door in one layout and no door in the other layout.

Input

The only line of input contains three integers n, m and k ($2 \leq n, m \leq 50; 1 \leq k \leq 1000$).

Output

Output a single integer, the number of valid door layouts, mod $10^9 + 7$.

Example

Standard Input	Standard Output
2 3 2	2
5 2 3	6

Problem I: Escaping the Beacon Field

In a high-security robotics testing facility, engineers are experimenting with an autonomous drone, flying at a fixed height. Looking directly from above, the drone appears as a triangle and can move freely in the plane, but it is not allowed to rotate — only translations (shifting without turning) are permitted.

Inside the testing area, there are three fixed laser beacons on the ground pointing directly upward. Each beacon is represented by a single point in the plane. These beacons form a security trap. If the drone (including its boundary) ever touches or crosses any beacon, the experiment fails. The drone initially occupies a given position, defined by the coordinates of its three vertices forming a triangle.

Your task is to determine whether the drone can escape to infinity — that is, move arbitrarily far away without ever colliding with any of the three beacons.

Input

The first line contains an integer t ($1 \le t \le 10\,000$), the number of test cases.

Each test case consists of two lines:

- The first line contains six integers x_1, y_1, x_2, y_2, x_3 and y_3 ($-10^9 \le x_1, y_1, x_2, y_2, x_3, y_3 \le 10^9$), the coordinates of the triangle's vertices. These three points define a non-degenerate triangle (the area is non-zero).
- Similarly, the second line contains six integers x_{b1}, y_{b1}, x_{b2}, y_{b2}, x_{b3} and y_{b3} ($-10^9 \le x_{b1}, y_{b1}, x_{b2}, y_{b2}, x_{b3}, y_{b3} \le 10^9$), the coordinates of the three distinct fixed beacon points.

It is guaranteed that none of the beacons are inside or on the border of the triangle.

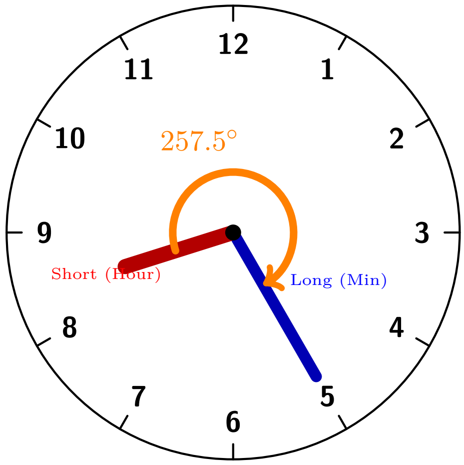
Output

Print exactly one line per test case. Print "`Yes`" if the drone can escape, Otherwise, print "`No`".

Example

Standard Input	Standard Output
4 0 0 1 0 0 1 2 2 3 3 4 4 5 5 3 2 2 4 5 4 2 3 4 5 5 5 3 2 2 4 5 4 2 3 3 5 5 5 3 2 2 4 4 3 2 3 3 5	Yes No Yes No

Problem J: Nightmare before the contest



It's the night before the ICPC 2025 contest, and you have been awake for days practicing to get the best results. Finally, you decide to get some rest, and at exactly 12:00 midnight, you go to sleep...

Di-di-di-di-di-di-diiiing!!! Di-di-di-di-di-di-diiiing!!!!!!!

You wake up in horror to the sound of an old analogue alarm clock. It's been a shock, and it's unlikely that you can go back to sleep anymore. So you are wondering how much you have slept.

The clock is missing all of its digits. It only contains a short-hand, denoting the hours, and a long-hand, denoting the minutes. The clock's hands don't move smoothly, instead, at every minute mark, the clock's short-hand instantly moves 0.5 degrees forward and the clock's long-hand instantly moves 6 degrees forward.

Note that it takes 1 hour for the long-hand to turn 360 degrees, and it takes 12 hours for the short-hand to turn 360 degrees.

Since the clock has no digits, the only way to figure out the time is to measure the angle between the two hands. The angle is measured starting from the short-hand, going clock-wise until reaching the long-hand.

Given this angle in degrees, find out how much you've managed to sleep. This time should be between "00:01" and "12:00", inclusive. It can be shown that it is always possible to find this time using the given angle.

Input

The only line of input contains a number between 0 and 359.5, the angle between the two clock hands. Note that this number is either an integer, or a floating point number ending in ".5".

Output

Print the sleeping time on a single line, in the format "`HH:MM`". Note that exactly 5 characters should be printed.

Example

Standard Input	Standard Output
`0`	`12:00`
`257.5`	`08:25`

Problem K: Mini Spreadsheet Engine

A research team is developing a lightweight spreadsheet engine for embedded systems where full-featured office software cannot be installed. Instead of a graphical interface, the system works purely with numerical tables and a limited set of formulas.

A spreadsheet is a table of cells arranged in rows and columns, with one-based indexing. Each cell is identified by its column letter(s) and row number (e.g., `B3` is column 2, row 3), and a range like `A1:C2` means the rectangle of cells from the top-left cell `A1` to the bottom-right cell `C2`, inclusive.

You are given a two-dimensional grid of numbers, representing the contents of a spreadsheet file. Each cell contains a single integer value, just like in Microsoft Excel, a well-known spreadsheet software.

After the table is given, the system receives a sequence of simple Excel-style commands, such as:

- `sum(A1:A10)`
- `sum(A10:D10)`

Each command specifies a function and a rectangular range of cells, and you must compute and return the result.

The supported functions are strictly limited to:

- `sum` — sum of all values in the range
- `min` — minimum value in the range
- `max` — maximum value in the range

The spreadsheet uses the following cell naming convention:

- Rows are numbered starting from 1 at the top: $1, 2, 3, \ldots$
- Columns are labeled alphabetically starting from A:
  - Columns 1–26: $A, B, C, \ldots, Z$,
  - Columns 27–52: $AA, AB, \ldots, AZ$,
  - Columns 53–78: $BA, BB, \ldots, BZ$,
  - The pattern continues with $CA, \ldots, ZZ$,
  - Column labels continues with 3 letters, then 4 letters, and so on.
- Each cell is referenced as `ColumnRow`, e.g., `A1`, `F234`, `AA5`.

Your task is to parse each command, interpret the referenced cell range, and compute the correct result.

Input

The input consists of three parts:

Spreadsheet size: the first line contains two integers R and C ($1 \leq R \times C \leq 10^5$), the number of rows and columns.

Spreadsheet values: the next R lines each contain C integers, each integer represents the value of a single spreadsheet cell. The absolute value of the given number is at most 10^9.

Excel-Style commands: the next line contains an integer Q ($1 \leq Q \leq 10^6$), the number of commands.

The next Q lines each contain a single command in the format:

- `function(cell1:cell2)`

Where:

- function is one of `sum`, `min`, `max`
- `cell1` and `cell2` define top-left and bottom-right cells of the range respectively
- Each `cell` is in Excel format (e.g., $A1$, $D10$, $AA25$)

Output

For each command, print one line containing the computed integer result. results must be in the same order as input commands.

Example

Standard Input	Standard Output
5 4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 4 sum(A1:A5) max(B2:C4) min(C3:D5) sum(A1:D1)	45 15 11 10
4 5 10 20 30 40 50 11 21 31 41 51 12 22 32 42 52 13 23 33 43 53 3 sum(A1:B2) max(A1:C3) min(B2:D4)	62 32 21

Problem L: Modular Fortress

You are a logistics engineer for a company that 3D prints modular fortresses. A design consists of P rectangular cuboid blocks that may overlap in space. The final printed structure S is the union of all these blocks. The coordinate system consists of unit blocks, where a unit block at (x, y, z) occupies the space $[x, x+1] \times [y, y+1] \times [z, z+1]$. A unit block is part of S if it is a part of at least one of the given cuboids.

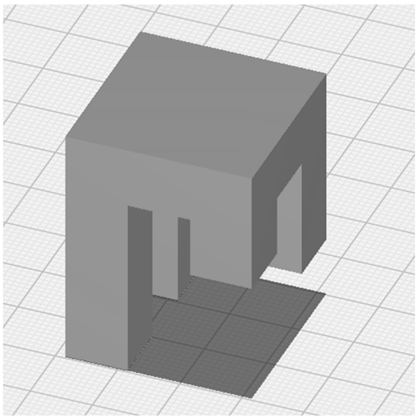
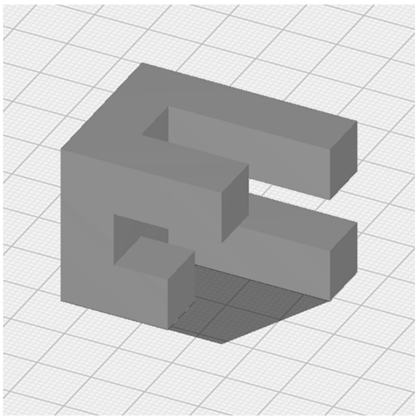
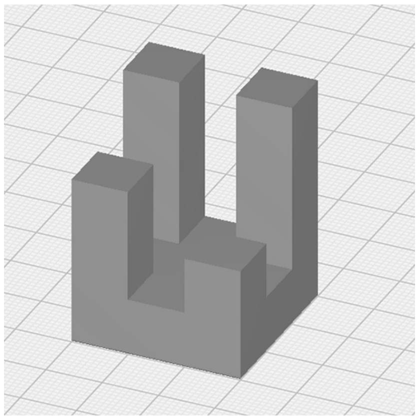
It is guaranteed that the final structure S forms a solid connected component (all blocks are connected to each other directly, or indirectly, via other blocks).

To minimize material costs, you can re-orient the entire design before printing. To re-orient the design, the entire design S can be rotated by any multiple of 90° around any of the X, Y, or Z axes.

After orientation, the structure is lowered vertically along the Z-axis until its lowest unit block(s) rest on the base plate ($z = 0$).

The printing process requires **Support Filament** to hold up overhanging parts. A unit block (x, y, z) at coordinates $z \geq 0$ must be filled with support filament if both of these conditions are satisfied:

1. It is not part of the structure ($(x, y, z) \notin S$).
2. There exists a unit block $(x, y, z') \in S$ in the same vertical column such that $z' > z$.

You must compute two values:

- F_{block}: The total volume (number of unique unit blocks) of the structure S.
- $\min F_{support}$: The minimum amount of support filament needed, considering all possible orientations.

Input

The first line contains one integer P ($1 \leq P \leq 50$).

The next P lines each contain six space-separated integers $x_1, y_1, z_1, x_2, y_2, z_2$ ($1 \leq x_1 \leq x_2 \leq 100$; $1 \leq y_1 \leq y_2 \leq 100$; $1 \leq z_1 \leq z_2 \leq 100$), describing the coordinates of a cuboid block.

Output

Output two space-separated integers, the constant F_{block} and the minimum possible $F_{support}$.

Example

Standard Input	Standard Output
5 1 1 1 1 1 4 3 1 1 3 1 3 1 3 2 1 3 4 3 3 3 3 3 4 1 1 4 3 3 4	18 0
4 1 1 1 5 5 2 1 1 5 5 5 6 1 1 1 2 2 6 4 4 1 5 5 6	116 12