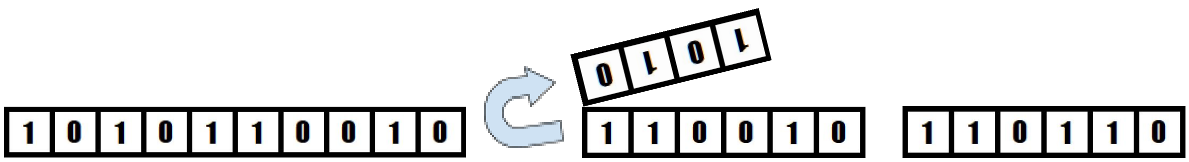
Problem A: Test and Binary Birthday

Filename: `birthday`
Time limit: `1 second`

Test Student received a binary string for his birthday! However, he doesn't like 0's in his binary strings. He wrote the string down on a strip of squares, and now he wants to convert the string to all 1's.

Test can take any continuous piece of the strip from either end and fold it over the rest of the strip, causing the two pieces to overlap. When the pieces overlap, if either of the overlapping squares contains a 1, he writes a 1 on the new combined strip. He wants as few 0's as possible. What is the smallest number of 0's he could have on the folded strip, assuming he folds it in the best place?



An example of folding a left-side strip over and replacing the values.

Input

Each test case will begin with a single line containing a single integer ***N*** (1 ≤ ***N*** ≤ 1000). The next line will contain a string of length ***N***, where each character is either '0' or '1', representing the binary string.

Output

Print a single integer, the minimum number of 0's Test can have on a well-folded strip.

Samples

Input	Output
`7` `1000001`	`2`
`11` `00111011001`	`1`

Problem B: Ducksort

Filename: `ducksort`
Time Limit: `2 seconds`

Trevor the Duck lives in a big park with a pond in the center. All around the pond are ***n*** benches numbered from ***1*** to ***n***. Trevor usually hangs out in the center of the pond, but when he is hungry he likes to see if he can get some easy free food from one of the park goers. Trevor has lived in the pond for a long time, and recently he has started keeping track of how many other ducks visit each bench for food and the expected amount of food distributed from each bench in a day.

Trevor plans to visit each bench in search of food, in a specified order. In particular, he prioritizes benches with more expected food. If multiple benches have the same amount of expected food, he picks the one visited by the fewest amount of ducks first. If two benches have the same statistics exactly, he visits the one with the lower number first.

Given a description for each of ***n*** benches, output the order in which Trevor will visit benches looking for food.

Input

The first line contains an integer, n ($1 \le n \le 10^5$), the number of benches around the pond. Two lines follow. The first line of these lines has ***n*** integers, the i^{th} of which is how much food is expected at the i^{th} bench. The next line also has ***n*** integers, the i^{th} of which is how many ducks visit the i^{th} bench. All of the integers are positive and at most $\mathbf{10^9}$.

Output

Output a permutation of the integers 1 to ***n*** representing the order in which Trevor will visit the benches.

Samples

Input	Output

`3`
`2 3 2`
`2 9 1`
	`2 3 1`

`3`
`1 1 1`
`3 1 2`
	`2 3 1`

Problem C: Jacob and Bottle Flips

Filename: `flip`
Time limit: `2 seconds`

On his trip to Death Valley, Jacob performed the lowest-elevation bottle flip of all time, 282 feet below sea level! Jacob did not want to stop there. In fact, he wanted to set a new record with a special 2D grid of bottles.

In each cell of the 2D grid Jacob has placed a bottle. Initially, all of the bottles stand upright. In the ith query, Jacob wants to flip the bottle at (x_i, y_i) f_i times, as well as the bottles edge-adjacent to it. The adjacent bottles can have coordinates (x_i+1, y_i), (x_i-1, y_i), (x_i, y_i+1), (x_i,y_i-1). The top left corner of the grid is the cell (1, 1) and the bottom right corner is (n, m).

When Jacob flips a bottle, if it is initially upright it will land bottom-up, and vice versa. Help Jacob determine how the board will look at the end.

Input

The first line contains two integers, n ($1 \le n \le 100$) and m ($1 \le m \le 100$), the number of rows in the grid and the number of columns, respectively. The next line contains an integer k($1 \le k \le 2*10^5$), the number of locations on the grid that Jacob wants to flip a bottle. k lines follow, each describing one flip location.

Each flip location description contains three positive integers: x ($1 \le x \le n$), y ($1 \le y \le m$), and f($1 \le f \le 10^{18}$), meaning that Jacob will perform f flips at location (x, y).

Output

Print n lines of m characters where the character at (x, y) is a 1 if the bottle is bottom-up and a 0 if it's upright, representing the final positions of each bottle in the grid.

Samples

Input	Output

`4 4`
`1`
`2 2 3`
	`0100`
	`1110`
	`0100`
	`0000`

`4 4`
`2`
`1 1 1`
`2 2 3`
	`1000`
	`0110`
	`0100`
	`0000`

Problem D: Tower Counting

Filename: `tower`
Time limit: `1 second`

Ian is playing with the Towers of Hanoi! The Towers of Hanoi is a logic puzzle involving rings stationed on pegs, as is shown in the photo. In a valid construction of the Towers of Hanoi, ***N*** distinct rings are placed on ***M*** pegs in such a way that no ring has a larger radius than the ring below it. Ian is curious about how many possible constructions exist for different values of ***N*** and ***M***. Help him find some answers!

Input

Each test case will consist of a single line with three integers ***N***, ***M***, and ***Z*** ($1 \le N, M, Z \le 10^9$). Compute the number of valid constructions of the Towers of Hanoi with ***N*** rings and ***M*** pegs. Since this number may be very large, print the answer modulo ***Z***[1].

Output

The number of valid constructions of the Towers of Hanoi with ***N*** rings and ***M*** pegs.

Samples

Input	Output
`2 3 101`	`9`
`3 2 6`	`2`

[1]*modulo*: computed with the % operator; (p % q) computes the remainder of p divided by q. For example: 8 % 3 = 2

Problem E: Extra Set

Filename: `extraset`
Time limit: `2 seconds`

Anya loves playing the game Set. The game consists of 81 cards. Each card has a picture with four attributes: number, color, shape and shading and there are three possible values for each attribute (number - {1, 2, 3}, color - {red, green, purple}, shape - {oval, diamond squiggle}, shading - {none, lines, solid}).

In the game, several cards are laid out and the first person to claim a set, wins that set. A set is three cards such that for each attribute, all three cards in the set either share the attribute or are all different for that attribute. For example, the cards [1, red, oval, lines], [2, red, oval, solid], [3, red, none] form a set because they have three different numbers, the same color, the same shape, and three different shadings.

Anya already regularly beats Arup, but Arup would like to train Anya to be even better in the game. He's thought of a harder version of the game where each card has k attributes, each of which has three possible values taken from the set {0, 1, 2}. Of the possible 3^k distinct cards, consider a set of n of these cards laid out. Arup would like Anya to be able to figure out how many combinations of three cards out of the n cards laid out form sets (a set of three cards where each attribute is either shared or different). Write a program to help Anya ace Arup's challenge!

Given the number of attributes for a Set card game, as well as the description of several cards from the game, determine the number of combinations of three cards from the given cards that form sets.

Input

The first line of input for will contain two space separated positive integers: $\boldsymbol{k}$ ($3 \le \boldsymbol{k} \le 19$), representing the number of attributes for the cards in the game, and $\boldsymbol{n}$ ($3 \le \boldsymbol{n} \le 1500$), representing the number of cards laid out for the game. The cards for the game follow, one per line. In particular, the i^{th} of these lines will contain $\boldsymbol{k}$ space separated integers $\boldsymbol{c_{i,1}}, \boldsymbol{c_{i,2}}, \boldsymbol{c_{i,3}}, \ldots, \boldsymbol{c_{i,k}}$ ($0 \le \boldsymbol{c_{i,j}} \le 2$), where $c_{i,j}$ represents the $\boldsymbol{j}^{th}$ attribute value of the $\boldsymbol{i}^{th}$ card. It is guaranteed that each card in a single game will be unique.

Output

On single line, output a single integer on a line by itself: the number of different combinations of three cards that form a set.

Samples

Input	Output
3 3 0 1 2 0 2 2 0 0 2	1
4 7 0 0 0 1 0 0 1 0 1 0 0 1 2 0 2 2 2 0 0 1 2 0 1 0 1 1 1 1	3

Problem F: Gauss's Detention

Filename: `gauss`
Time limit: `1 second`

Gauss and his class misbehaved, so his teacher gave the class the task of adding the first 100 integers. Gauss thwarted his teacher's intention by discovering a formula for the sum and avoided doing any addition! Now, his teacher wants revenge! He's figured out that Gauss doesn't like adding large numbers. In fact, if he's adding two numbers a and b, it takes him $a+b$ ms. Thus, his teacher has decided to give Gauss a random string of numbers with no pattern, to add. For example, if Gauss had to add 137, 213, 98 and 49, he could add 137+213 = 350, then add 98 + 49 = 147 and finally add 350 + 147 = 497. The total amount of time this would take him if he added the numbers in this order is 350 ms + 147 ms + 497 ms = 994 ms. It turns out, it would have been better if he added 98 + 49 = 147, 147 + 137 = 284 and 284 + 213 = 497, which would have taken him 147 ms + 284 ms + 497 ms = 928 ms. Given a list of numbers Gauss has been asked to add, determine the minimum amount of time it will take him in ms to calculate the sum, assuming he adds up the numbers in the optimal order.

Input

The first line of input will contain a single positive integer $\boldsymbol{n}$ ($\boldsymbol{n} \le 30000$), representing the number of positive integers Gauss has to add for the input case.

The second line of each input case will contain the $\boldsymbol{n}$ space separated integers:
$\boldsymbol{a_i}$ ($1 \le i \le \boldsymbol{n}$, $1 \le \boldsymbol{a_i} \le 4000$) that Gauss must add for the input case.

Output

For each input case, on a line by itself, output the minimum amount of time in ms that it will take Gauss to complete the addition posed to him.

Samples

Input	Output
`3` `1 2 4`	`10`
`4` `137 213 98 49`	`928`
`8` `1 4 9 16 25 36 49 64`	`512`

Problem G: Phil and Plinko

Filename: `plinko`
Time limit: `4 seconds`

The carnival's in town, and Phil the Philosopher has gone to play some games on the midway. He decides to play a game similar to Plinko. Phil has philosophized that, unlike most carnival games, this one can be won deterministically as long as he makes the right choices.

The Plinko game is made up of a grid of several dials and blocks that control the path of a small ball. When a ball falls onto a dial, depending on what direction the dial is facing, it can travel to any of the three spaces closest to it on the row below. The ball can not travel into any space that contains a block, and it will come to rest if it's directed towards one. Phil is allowed to change the direction of up to K of the dials on the grid before dropping a ball onto any dial on the grid.

Each of the spaces on the grid containing a dial have a score. If the ball travels through a space before falling off the grid or coming to rest, that space's value is added to Phil's score. If Phil turns the dials optimally and drops the ball in the right place, what is the best score he can get?

Input

The first line of input will contain three positive integers: $\boldsymbol{N}$, $\boldsymbol{M}$ and $\boldsymbol{K}$ ($\boldsymbol{N},\boldsymbol{M} \leq 10^5$, $\boldsymbol{N} \cdot \boldsymbol{M} \leq 10^5$, $\boldsymbol{K} \leq 200$), representing the number of rows in the grid, the number of columns in the grid, and the maximum number of dials Phil can change. Following this are $\boldsymbol{N}$ lines of $\boldsymbol{M}$ characters representing the original state of the Plinko grid. The characters **/**, **|**, and **** correspond to dials causing the ball to move down/left, straight down, and down/right respectively. **#** represents a block. Following this are $\boldsymbol{N}$ lines, each containing $\boldsymbol{M}$ space separated integers $\boldsymbol{a}_{i,j}$ ($0 \leq \boldsymbol{a}_{i,j} \leq 10^4$), representing the score of the dial in the j^{th} cell of the i^{th} row.

Output

On a line by itself, output a single integer: the highest score Phil can get.

Samples

<table>
<tr><th>Input</th><th>Output</th><th>Explanation</th></tr>
<tr><td><code>4 4 2
\//|
#\||
\#\/
|//\
3 6 4 3
0 4 5 3
9 0 7 3
4 7 4 6</code></td><td><code>26</code></td><td><code>\//| \|/|
#\|| #/||
\#\/   \#\/
|//\ |//\</code>

<code>The best path for the ball is highlighted above. The right side shows the grid after changes to 2 dials.</code></td></tr>
</table>

Problem H: Purple Rain

Filename: `purplerain`
Time limit: `4 seconds`

Purple rain falls in the magic kingdom of Linearland which is a straight, thin peninsula. On close observation however, Prof. Nelson Rogers finds that actually it is a mix of Red and Blue drops. In his zeal, he records the location of each of the raindrops to fall with its corresponding color in different locations along the peninsula. He wants to answer the following question: which section of Linearland had the least purple rain? That is, which section had the greatest difference between red rain and blue rain?

After some thought, he decides to model the problem as follows: Divide the peninsula into ***n*** sections and describe it as a sequence of ***R*** or ***B*** values depending on whether the rainfall in that section is primarily red or blue. Then, find the part consisting of consecutive sections where the absolute difference of the count of ***R***s and ***B***s is maximized.

Input

The input will consist of a single line with a string ***s*** ($1 \le |s| \le 100{,}000$), where every character in ***s*** is either a capital ***R*** or a capital ***B***. This string describes the peninsula, from west to east.

Output

Output two integers, indicating the start and end of the part of the peninsula which maximizes the difference between ***R***s and ***B***s. The first character of ***s*** is at position 1, and the last is at position ***n***. Output the smaller index first. If there are multiple parts that feature the same maximal absolute difference, print the one with the smallest starting position. If there are multiple such parts starting at that same smallest starting position, print the shortest of those.

Samples

Input	Output
`BBRRBRRBRB`	`3 7`
`BBRBBRRB`	`1 5`