IMC Coding Competition 2024 Problems

UNSW CPMSoc and UNSW CSESoc

13 March 2024

Contents

Minnemax [Beginner Division only]

Program time limit: 1 second
Program memory limit: 512 MB

Write a program, which, given two integers a and b, evaluates $\min(a, b) + \max(-a, -b)$.

Here, $\min(a, b)$ is the minimum of a and b, and $\max(a, b)$ is the maximum of a and b.

Input

The first and only line of input contains the two integers a and b.

You should read from standard input.
In Python, you could use the line `a, b = map(int, input().split())`.
In C or C++, you could use the line `int a, b; scanf("%d%d", &a, &b);`.

Constraints

Each input case will satisfy the following constraints:

- $-1000 \leq a, b \leq 1000$.

Output

Output a single integer, the value of $min(a, b) + max(-a, -b)$.

You should write to standard output.
In Python, you could use the line `print(answer)`.
In C or C++, you could use the line `printf("%d\n", answer);`.

Sample Input 1

```
0 0
```

Sample Output 1

```
0
```

Scoring

Your program will be run on 10 secret test cases one after another, each worth 10% of the points. Recall that your final score on the task is the score of your highest scoring submission.

Tim Estable [Beginner Division only]

Program time limit: 1 second
Program memory limit: 512 MB

Tim Estable has been learning his times tables. Today, he tried to write down all the times tables up to N.

Given Tim's grid of answers, how many of them are correct?

Input

The first line of input contains the integer N, which is the size of Tim's grid.
N lines follow, each containing N integers. This is Tim's grid of answers.

You should read from standard input.
In Python, you could use the line `N = int(input()); grid = [list(map(int, input().split())) for i in range(N)]`.
In C or C++, you could use the line `int N; scanf("%d", &N); int grid[N][N]; for (int i = 0; i < n; i++) for (int j = 0; j < n; j++) scanf("%d", &grid[i][j]);`.

Constraints

For all test cases:

- $1 \leq N \leq 500$.
- Each value in Tim's grid of answers is between 1 and 1000000, inclusive.

Additionally:

- For 50% of test cases, $N \leq 5$.
- For 50% of test cases, there are no additional constraints.

Output

You should write to standard output.
In Python, you could use the line `print(answer)`.
In C or C++, you could use the line `printf("%d\n", answer);`.

Sample Input 1

```
3
1 2 3
2 3 4
3 4 6
```

Sample Output 1

```
5
```

Explanation 1

In this case, the last two entries of the last two rows are wrong. Tim got 5 answers correct.

Sample Input 2

```
5
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
```

```
4 8 12 16 20
5 10 15 20 25
```

Sample Output 2

```
25
```

Explanation 2

In this case, all of Tim's answers are correct. Well done Tim!

Scoring

Your program will be run on 4 secret test cases one after another, each worth 25% of the points. Recall that your final score on the task is the score of your highest scoring submission.

Glowing Trees [Beginner Division only]

Program time limit: 1 second
Program memory limit: 512 MB

In a magical forest, there are mystical trees known as "glowing trees". These trees emit light in a peculiar pattern that forms a pyramid shape.

Write a program which outputs the pattern produced by a glowing tree, given its size N. See the sample cases for examples.

Input

The first and only line of input contains the integer N, the size of the glowing tree.

You should read from standard input.
In Python, you could use the line `N = int(input())`.
In C or C++, you could use the line `int N; scanf("%d", &N);`.

Constraints

For all test cases:

- $1 \leq N \leq 100$.

Output

Output the pattern of the glowing tree. The height of the trunk should be one more than the result of dividing N by 3 (rounding down).

You should write to standard output.

Sample Input 1

```
5
```

Sample Output 1

```
....*....
...***...
..*****..
.*******.
*********
....*....
....*....
```

Explanation 1

This is the pattern of a glowing tree (and trunk) of size 5. Note that the trunk has height $\lfloor 5/3 \rfloor + 1 = 2$.

Sample Input 2

```
10
```

Sample Output 2

```
.........*.........
........***........
.......*****.......
......*******......
.....*********.....
....***********....
...*************...
..***************..
.*****************.
*******************
.........*.........
.........*.........
.........*.........
.........*.........
```

Explanation 2

This is the pattern of a glowing tree of size 10.

Scoring

Scoring

Your program will be run on 10 secret test cases one after another, each worth 10% of the points. Recall that your final score on the task is the score of your highest scoring submission.

Tringle

Program time limit: 1 second
Program memory limit: 512 MB

Trixie loves her three sided polygons. Whenever she sees three numbers next to each other, she will try to make a triangle with those numbers as its side lengths. She then screams out what type of triangle she just made, although sometimes she cannot make a triangle.

Given any three numbers, what type of triangle should we expect Trixie to scream?

Input

The first and only line of input contains the three integers a, b and c.

You should read from standard input.
In Python, you could use the line `a, b, c = map(int, input().split())`.
In C or C++, you could use the line `int a, b, c; scanf("%d%d%d", &a, &b, &c);`.

Constraints

Each input case will satisfy the following constraints:

- $1 \leq a, b, c \leq 2000$.

Output

Output either `EQUILATERAL`, `SCALENE`, `RIGHT ANGLED`, `ISOSCELES` or `NOT A TRIANGLE`, based on the type of triangle formed by the three side lengths. If the triangle is simultaneously right angled and scalene, print `RIGHT ANGLED` only.

You should write to standard output.
In Python, you could use the line `print("EQUILATERAL")`.
In C or C++, you could use the line `printf("EQUILATERAL\n");`.

Sample Input 1

```
5 5 5
```

Sample Output 1

```
EQUILATERAL
```

Sample Input 2

```
3 6 7
```

Sample Output 2

```
SCALENE
```

Sample Input 3

```
3 4 5
```

Sample Output 3

```
RIGHT ANGLED
```

Scoring

Your program will be run on 20 secret test cases one after another, each worth 5% of the points. Recall that your final score on the task is the score of your highest scoring submission.

Insertion

Program time limit: 1 second
Program memory limit: 512 MB

You can *insert* a string X into a string Y by splitting Y into two parts (one of which may be empty), and placing X in between those two parts.

For example, you can insert `train` into `sage` to get `strainage`, since `sage` can be split into the two parts `s` and `age`, and then `train` can be placed in between them to obtain `s` + `train` + `age` = `strainage`.

Given two strings X and Y of lowercase letters, determine whether there exists a string that you can insert into X to obtain Y.

Input

The first line of input contains two integers $|X|$ and $|Y|$, the length of X and the length of Y respectively.
The second line contains the string X of $|X|$ lowercase letters.
The second line contains the string Y of $|Y|$ lowercase letters.

You should read from standard input.

Constraints

Each input case will satisfy the following constraints:

- $1 \leq |X| \leq |Y| \leq 100\,000$.

Additionally:

- For Subtask 1 (30% of points), $|X| = 1$
- For Subtask 2 (50% of points), $Y \leq 1000$.
- For Subtask 3 (20% of points), there are no additional constraints.

Output

If you can insert a string into X to obtain Y, output `YES`.
Otherwise, output `NO`.

You should write to standard output.

Sample Input 1

```
4 9
sage
strainage
```

Sample Output 1

```
YES
```

Explanation 1

In this case, you can insert the string `train`.

Sample Input 2

```
3 6
khi
kimchi
```

Sample Output 2

```
YES
```

Explanation 2

In this case, you can insert the string `imc`.

Sample Input 3

```
5 7
catan
caravan
```

Sample Output 3

```
NO
```

Explanation 3

In this case, there is no string you can insert.

Scoring

For each subtask (worth 30%, 50% and 20% of points, as per the Constraints section), your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves that subtask. Your program will receive the points for each subtask it solves. Recall that your final score on the task is the score of your highest scoring submission.

Parcels

Program time limit: 1 second
Program memory limit: 512 MB

Tim runs the most popular post agency in town and specialises in delivering only one single type of parcel, the infamous two by one parcel (its dimensions are 2 units by 1 unit). People come and go everyday, dropping parcels into the delivery chute which is 7 units wide. However, to Tim's annoyance, some people drop the parcels vertically (represented as a type 1 parcel in the input) while the others drop the parcels horizontally (represented as a type 2 parcel in the input).

In order to optimise the delivery schedule, Tim wants to see what the delivery chute looks like. So he's asked you to write him a program which, given where and how N parcels are dropped, prints out the final configuration of the delivery chute from top to bottom.

Since his delivery chute is infinitely tall, you will be required to print out the state of the delivery chute from the top row which contains a parcel all the way down to the bottom. Each space within the chute that contains a parcel will be represented as `*`, while each space in the parcel that does not contain a parcel will be represented as `-`.

Input

The first line of input contains the integer N, the number of parcels being dropped.
N lines follow, the ith of which contains the two integers t_i and c_i, where t_i represents the type of parcel that is dropped, and c_i is the column in which the parcel is dropped. If $t_i = 1$, a vertical parcel is dropped in column c_i. If $t_i = 2$, a horizontal parcel is dropped in columns c_i and $c_i + 1$.

You should read from standard input.

Constraints

For all test cases:

- $1 \leq N \leq 100\,000$.
- $1 \leq t_i \leq 2$, for all i.
- $1 \leq c_i \leq 7$, for all i.
- $t_i + c_i \leq 8$, for all i.

Additionally:

- For Subtask 1 (70% of points), $N \leq 1000$.
- For Subtask 2 (30% of points), there are no additional constraints.

Output

Output a grid of `*`s and `-`s, representing the final configuration of the delivery chute. The first row should be the topmost row containing a parcel.

You should write to standard output.

Sample Input 1

```
4
1 1
2 1
2 2
2 4
```

Sample Output 1

```
-**----
**-----
*------
*--**--
```

Explanation 1

A vertical parcel is dropped into the first column.
A horizontal parcel is then dropped in the first and second columns.
A horizontal parcel is then dropped in the second and third columns.
A horizontal parcel is then dropped in the fourth and fifth columns.

Scoring

For each subtask (worth 70% and 30% of points, as per the Constraints section), your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves that subtask. Your program will receive the points for each subtask it solves. Recall that your final score on the task is the score of your highest scoring submission.

Weird Numbers

Program time limit: 1 second
Program memory limit: 512 MB

Tim Estable hates maths. Learning his times tables was too hard. Tim Estable realised however that most numbers were easy to deal with for him, but he only had trouble with certain **weird** numbers.

A positive integer is a **weird** number if and only if there exist two identical adjacent digits in it. For example,

- 119 is weird.
- 445755 is weird.
- 1 is not weird.
- 123 is not weird.

Tim Estable gives you a single integer N, and asks you to find the smallest **non-weird** integer that is strictly greater than N. Seeing as Tim Estable cannot even recite his times tables, it's now your job to help him.

Input

The first and only line of input contains the integer N.

You should read from standard input.

Constraints

For all test cases:

- $1 \leq N \leq 10^{18}$.

Additionally:

- For Subtask 1 (30% of points), $1 \leq N \leq 100\,000$.
- For Subtask 2 (70% of points), there are no additional constraints.

Output

Print the smallest integer strictly greater than N which is **not** a weird number.

You should write to standard output.

Sample Input 1

```
343
```

Sample Output 1

```
345
```

Explanation 1

The smallest integer that is strictly greater than 343 is 344. Since 344 is a weird number, but 345 is not a weird number, the answer is 345.

Scoring

For each subtask (worth 30% and 70% of points, as per the Constraints section), your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves that subtask. Your program will receive the points for each subtask it solves. Recall that your final score on the task is the score of your highest scoring submission.

Nuts [Beginner Division only]

Program time limit: 1 second
Program memory limit: 512 MB

Three squirrels stumble across a large pile of nuts. After meticulously analysing the pile, they find that there are a total of n nuts.

They decide to split the nuts in the following way. The first squirrel takes 1 nut, then the squirrels take turns going around in a circle, taking 1 more nut than the previous squirrel. If the number of nuts remaining is no more than the number the squirrel should take, it takes all the nuts and they finish splitting.

How many nuts does the last squirrel who took any nuts have in total?

Input

The first and only line of input contains the integer n, the total number of nuts.

You should read from standard input.
In Python, you could use the line `N = int(input())`.
In C or C++, you could use the line `long long N; scanf("%lld", &N);`.
Note that if you are using C or C++, you may require `long longs`.

Constraints

For all test cases:

- $1 \leq N \leq 5 \times 10^{17}$.

Additionally:

- For Subtask 1 (50% of points), $N \leq 1\,000\,000$.
- For Subtask 2 (50% of points), there are no additional constraints.

Output

Output a single integer, the total amount of nuts acquired by the last squirrel to take any nuts.

You should write to standard output.
In Python, you could use the line `print(answer)`.
In C or C++, you could use the line `printf("%lld\n", answer);`.
Note that if you are using C or C++, you may require `long longs`.

Sample Input 1

```
12
```

Sample Output 1

```
4
```

Explanation 1

The turns go as follows:
The first squirrel takes 1 nut.
The second squirrel takes 2 nuts.
The third squirrel takes 3 nuts.
The first squirrel takes 4 nuts.
The second squirrel takes 2 nuts.
Squirrel 2 is the last one to take nuts, and has a total of $2 + 2 = 4$ nuts.

Scoring

For each subtask (worth 50% and 50% of points, as per the Constraints section), your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves that subtask. Your program will receive the points for each subtask it solves. Recall that your final score on the task is the score of your highest scoring submission.

Paper [Advanced Division only]

Program time limit: 1 second
Program memory limit: 512 MB

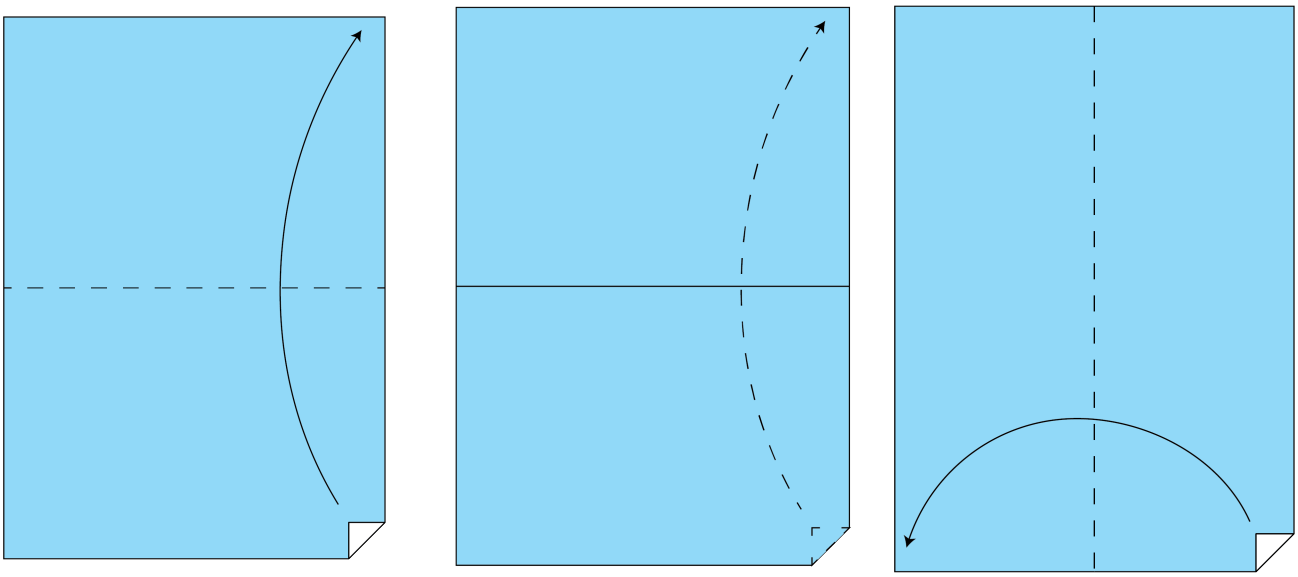
One day, you continually fold a piece of paper in half and in half again, until you stop and unfold the paper to its original size. You see an R by C grid of rectangles, with each rectangle surrounded creases. The vertical creases form their own R by $C-1$ grid, and the horizontal creases form their own $R-1$ by C grid.

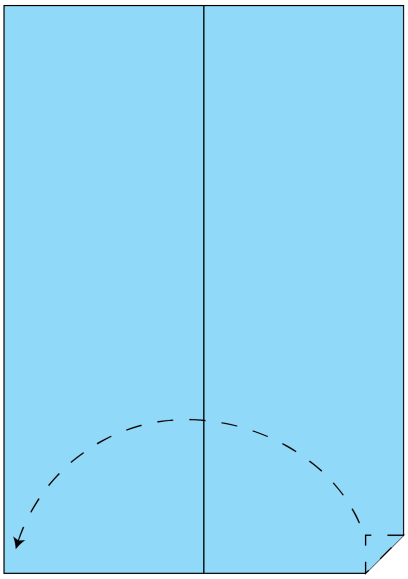
You wonder if you saw someone else's unfolded paper whether you could determine the folds they used to create it. Given whether each crease is a valley or a mountain, find a sequence of folds that could have created resulted in those creases, or determine that no such sequence exists.

There are two different fold directions, Up and Left, and two different fold types, Valley and Mountain. Therefore all folds in consideration are of one of the following types:

- Up Valley (`UV`). This fold is performed by taking the bottom of the paper and folding it towards you and up onto the front facing side. This would result in a horizontal valley crease when unfolded.
- Up Mountain (`UM`). This fold is performed by taking the bottom of the paper and folding it away from you and up onto the back facing side. This would result in a horizontal mountain crease when unfolded.
- Left Valley (`LV`). This fold is performed by taking the right side of the paper and folding it towards you and left onto the front facing side. This would result in a vertical valley crease when unfolded.
- Left Mountain (`LM`). This fold is performed by taking the right side of the paper and folding it away from you and left onto the back facing side. This would result in a vertical mountain crease when unfolded.

The diagrams below depict an Up Valley fold, Up Mountain fold, Left Valley fold and Left Mountain fold respectively.





Input

The first line of input contains the two integers R and C, which is the size of grid of rectangles formed by the creases.
R lines follow, each containing $C-1$ integers. This is the grid of vertical creases, where 0 represents a valley crease and 1 represents a mountain crease.
$R-1$ lines follow, each containing C integers. This is the grid of horizontal creases, where 0 represents a valley crease and 1 represents a mountain crease.

You should read from standard input.

Constraints

For all test cases:

- $2 \leq R, C \leq 1024$.
- R and C are both powers of 2.
- Every value in each grid is either 0 or 1.

Output

If there is no sequence of folds that could have created resulted in the given creases, output `NO`.
Otherwise, on the first line, output `YES`, and on each of the following lines, output either `UM`, `UV`, `LM`, or `LV`, specifying the sequence of folds.

You should write to standard output.

Sample Input 1

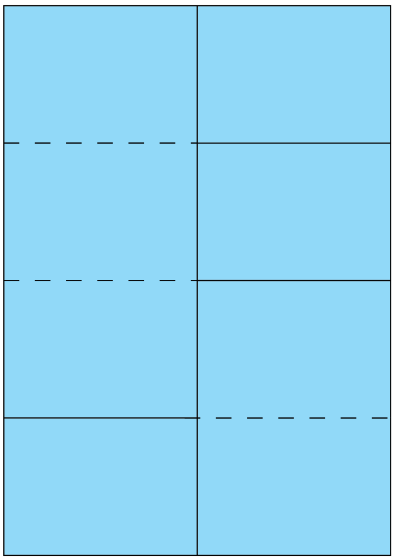
```
4 2
1
1
1
1
0 1
0 1
1 0
```

Sample Output 1

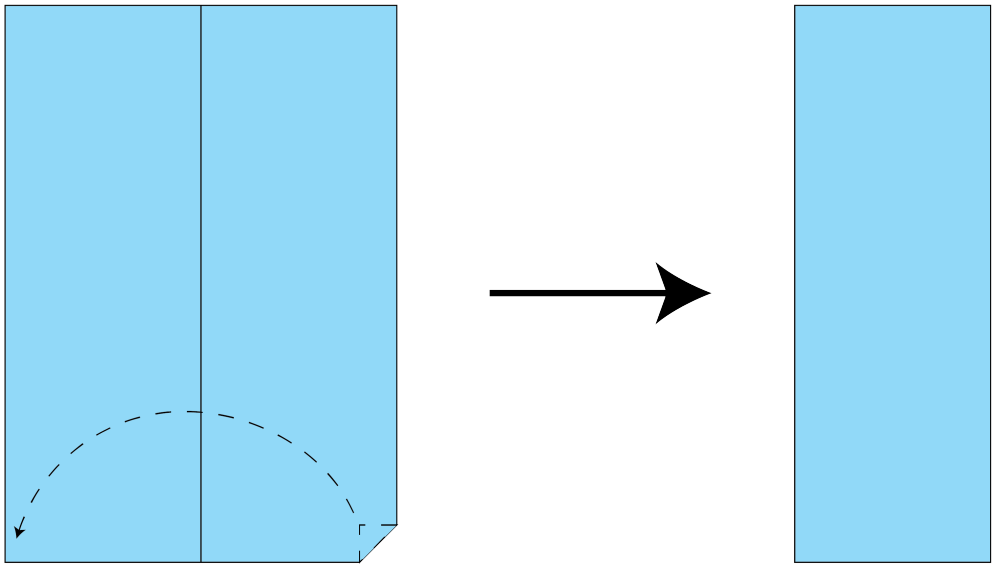
```
YES
LM
UV
UV
```

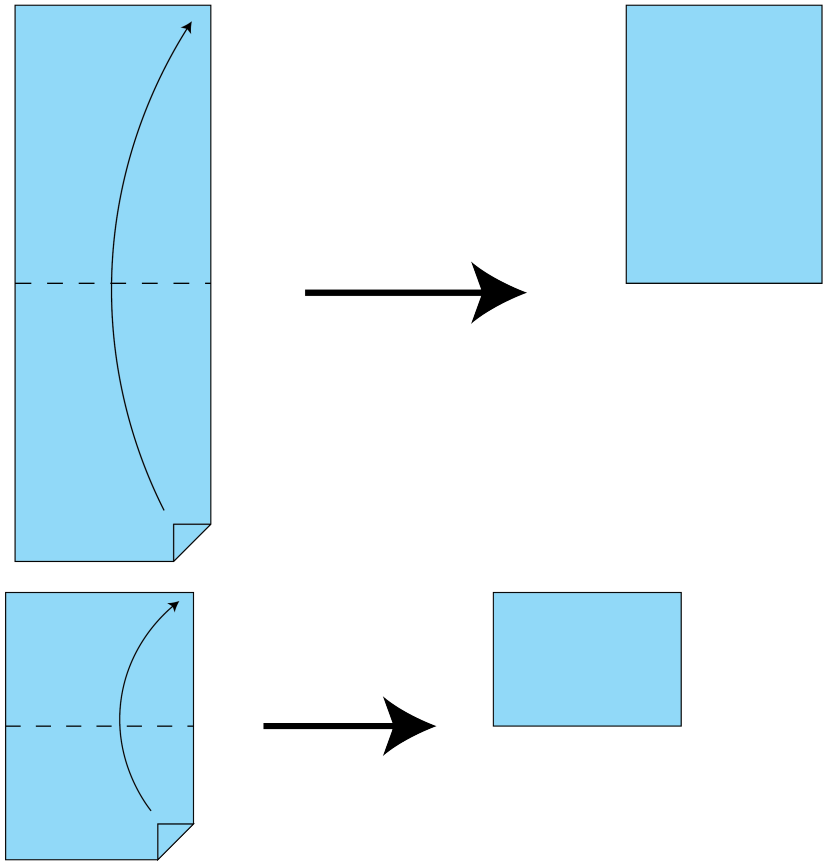
Explanation 1

The given creases are depicted below.



A sequence of folds which would result in those given creases is depicted below.





Sample Input 2

```
4 4
0 0 1
1 0 1
0 1 1
1 1 0
0 0 1 1
1 1 1 1
1 1 0 0
```

Sample Output 2

```
NO
```

Explanation 2

In this case, it can be shown that no sequence of folds results in the given creases.

Scoring

Your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves the task.

Seven Bag

Program time limit: 1 second
Program memory limit: 512 MB

In most modern releases of the video game Tetris™, the method the game uses to choose the next piece for the player is done using what is known as a "7-bag randomiser". This means that once the game starts, a "bag" containing 1 copy of each of the 7 pieces will be randomised, and this decides the order for your next 7 pieces. Each time a bag is emptied, a new bag is randomised, and this repeats itself until the game ends.

However in some versions of Tetris™, in order to increase the level of variation, it will use a 14-bag randomiser. This is similar with the only difference that every 14 pieces will contain exactly 2 copies of each of the 7 pieces. Similarly, a 21-bag randomiser, a 28-bag randomiser, and so on are all possible.

You are given a sequence of Tetris™ pieces received from the start of the game. What is the smallest bag size that could have been used to generate the Tetris™ pieces?

The 7 pieces are referred to by letters that each resembles: `I`, `O`, `T`, `J`, `S`, `Z` and `L`.

Input

The first line of input contains the integer N, the number of Tetris™ pieces that you receive.
The second line contain a string of N characters, describing the Tetris™ pieces that you receive.

You should read from standard input.

Constraints

For all test cases:

- $1 \leq N \leq 300\,000$.
- Each character of the string is either `I`, `O`, `T`, `J`, `S`, `Z` or `L`.

Additionally:

- For Subtask 1 (30% of points), $N \leq 1000$.
- For Subtask 2 (70% of points), there are no additional constraints.

Output

Output a single integer, the smallest possible bag size that could have produced the given sequence of Tetris™ pieces

You should write to standard output.

Sample Input 1

```
9
TSJZLIOTL
```

Sample Output 1

```
7
```

Explanation 1

We can have a 7-bag randomiser, where the first 7 letters belong to the first bag and the last 2 letters belong to the second bag.

Sample Input 2

```
10
TTTOJSZSLS
```

Sample Output 2

```
21
```

Explanation 2

Since we have three `T`s in a row at the start, they must all belong to the same bag, and so the smallest bag size has to be at least 21. In fact, 21 is the smallest possible bag size.

Scoring

For each subtask (worth 30% and 70% of points, as per the Constraints section), your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves that subtask. Your program will receive the points for each subtask it solves. Recall that your final score on the task is the score of your highest scoring submission.

Merchandise

Program time limit: 2 seconds
Program memory limit: 512 MB

CSESoc wants CPMSoc to sell some of their merch at the CPMShop. However, CPMSoc *doesn't* want anything sold, so they will try to drive away every customer by pricing the items highly.

The N merchandise items will be placed from left to right along a line, numbered from 1 to N. Then C customers will enter the shop, and the ith customer will examine item 1 up to item c_i (inclusive), and will choose not buy anything if and only if the sum of their prices is at least p_i.

The higher the sum of squares of item prices, the more suspicious CSESoc will be. What is the minimum possible sum of squares of item prices, such that no customer will buy anything?

Input

The first line of input contains the two integers N and C, the number of items and the number of customers respectively.
C lines follow, the ith of which contains the two integers c_i and p_i, indicating that the sum of prices of the first c_i items should be at least p_i.

Constraints

For all test cases:

- $1 \leq N, C \leq 100\,000$.
- $1 \leq c_i, p_i \leq 10^5$ for all i.
- $c_i < c_{i+1}$ for all $i < N$.

Additionally:

- For Subtask 1 (10% of points), $C = 1$.
- For Subtask 2 (20% of points), $p_i \leq c_i$ for all i.
- For Subtask 3 (20% of points), $M, C \leq 1000$.
- For Subtask 4 (50% of points), there are no additional constraints.

Output

Output the minimum possible total suspicion.

You should write to standard output.

Sample Input 1

```
5 2
2 4
3 5
```

Sample Output 1

```
9
```

Explanation 1

To drive away the first customer, we need the sum of the prices of first and second items to be at least 4. To drive away the second customer, we need the sum of the prices of the first three items to be at least 6. To satisfy this, the five items can have costs $[2, 2, 1, 0, 0]$. The sum of squares of item prices is $2^2 + 2^2 + 1^2 + 0^2 + 0^2 = 9$ which is as small as possible.

Scoring

For each subtask (worth 10%, 20%, 20% and 50% of points, as per the Constraints section), your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves that subtask. Your program will receive the points for each subtask it solves. Recall that your final score on the task is the score of your highest scoring submission.

Buildings [Advanced Division only]

Program time limit: 1 second
Program memory limit: 512 MB

You have N buildings in a row, numbered 1 to N from left to right. The ith building has a height of $A[i]$ metres.

Oh no! You forgot that the buildings are meant to be arranged in non-decreasing height order, from left to right. This means that for every pair of adjacent buildings, the height of the left building should be less than or equal to the height of the right building.

Fortunately, you can add 1 metre, or subtract 1 metre, from any building's height, for the low price of 1 dollar. You can repeat this operation any number of times, possibly more than once on the same building.

What is the minimum number of dollars you need to spend so that the buildings are arranged in non-decreasing height order?

Input

The first line of input contains the integer N, which is the number buildings.
The second line contains N integers $A_1, \ldots, A_N$, which are the heights of the buildings in metres.

You should read from standard input.

Constraints

For all test cases:

- $1 \leq N \leq 100\,000$.
- $1 \leq A[i] \leq 10^9$, for all i.

Additionally:

- For Subtask 1 (20% of points), $A[i] \geq A[i+1]$ for all $i < N$.
- For Subtask 2 (30% of points), $N \leq 1000$.
- For Subtask 3 (50% of points), there are no additional constraints.

Output

Output a single integer, the minimum number of dollars you need to spend so that the buildings are arranged in non-decreasing height order.

You should write to standard output.

Sample Input 1

```
5
1 2 6 5 2
```

Sample Output 1

```
4
```

Explanation 1

In this case, you could do the following:

- add 1 metre to building 1,
- subtract 1 metre from building 3, and
- add 3 metre to building 5.

The final heights are $[2, 2, 5, 5, 5]$, which is non-decreasing. This costs 4 dollars, which is the minimum possible.

Sample Input 2

```
4
8 8 3 3
```

Sample Output 2

```
10
```

Explanation 1

In this case, you could subtract 5 metres from the first two buildings.

The final heights are $[3, 3, 3, 3]$, which is non-decreasing. This costs 10 dollars, which is the minimum possible.

Scoring

For each subtask (worth 20%, 30% and 50% of points, as per the Constraints section), your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves that subtask. Your program will receive the points for each subtask it solves. Recall that your final score on the task is the score of your highest scoring submission.

Tim's Dance [Advanced Division only]

Program time limit: 2 seconds
Program memory limit: 512 MB

The night is young and Tim Estable is seventeen, so he's set to the dance floor. It's covered by a mirage of square lights in an R by C grid, with a light at position (r, c) for all $1 \leq r \leq R$ and $1 \leq c \leq C$.

To spice things up, there are N switches. Flicking the ith switch will toggle the state of every light in row r_i and column c_i **except** for the light at position (r_i, c_i). That is, if such a light was originally off, it will now turn on, and vice versa.

Tim Estable wants maximal jiving capabilities, so he wishes to calculate the total number of dance floor light configurations that are possible with the switches. All lights and switches are initially off. Note that configurations are only differentiated by which lights are on or off, not the switch states themselves. Since these numbers get big, Tim Estable wants the number modulo $10^9 + 7$.

Input

The first line of input contains the three integers R, C and N, which are the number of rows, the number of columns and the number of switches respectively.
N lines follow, the ith of which contains the two integers r_i and c_i.

Constraints

For all test cases:

- $1 \leq R, C, N \leq 10^5$.
- $1 \leq r_i \leq R$ for all i.
- $1 \leq c_i \leq C$ for all i.

Additionally:

- For Subtask 1 (30% of points), $R, C \leq 100$ and $N \leq 10$.
- For Subtask 2 (70% of points), there are no additional constraints.

Output

Output the total number of dance floor light configurations, modulo $10^9 + 7$.

You should write to standard output.

Sample Input 1

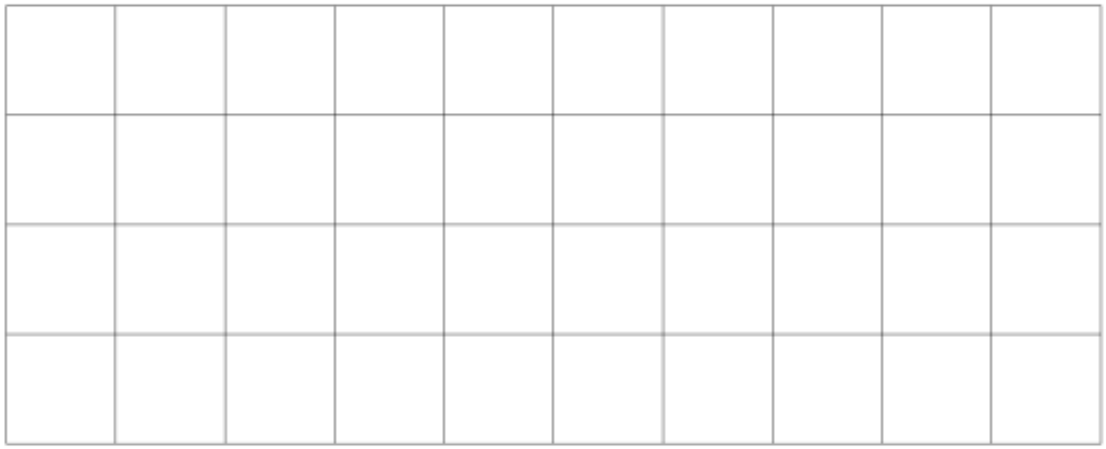
```
4 10 2
2 1
4 8
```

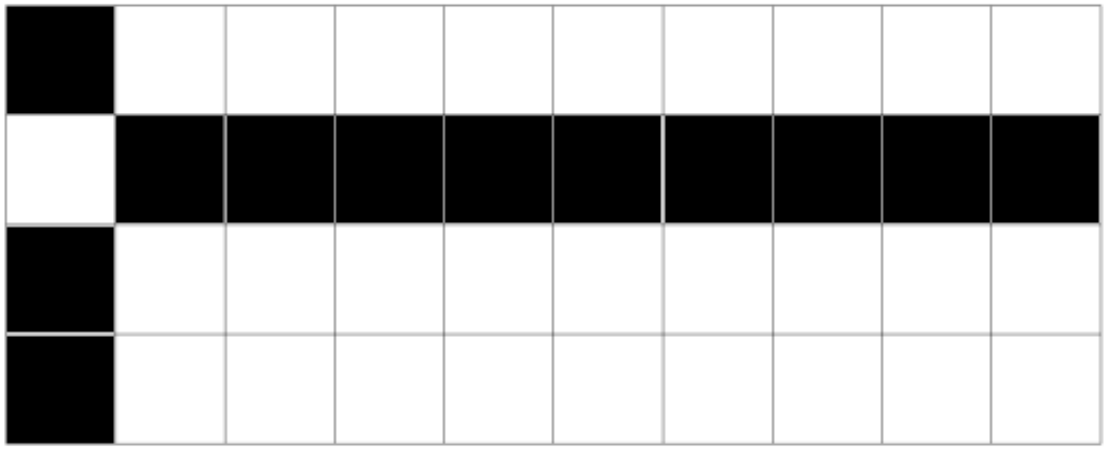
Sample Output 1

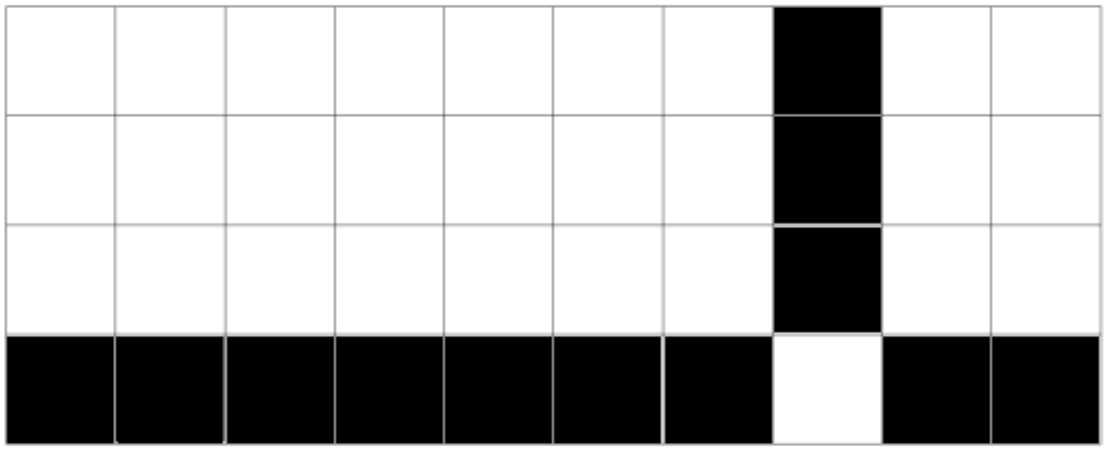
```
4
```

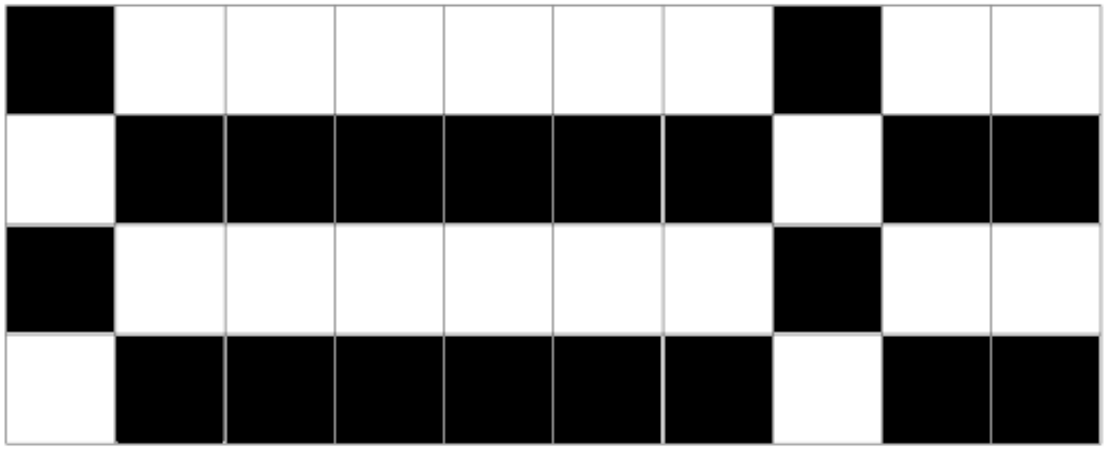
Explanation 1

The four possible dance floors in this case are depicted below.









IMC Banner V [Advanced Division only]

Program time limit: 3 seconds
Program memory limit: 512 MB

After extensive market research, it has been determined that the effectiveness of a company banner is solely determined by the number of times the company's name appears on it. In fact, there is still an effect even if there are other characters in between, as long as the letters of the company's name are evenly spaced out.

You are given a banner, which is a string made up of only capital letters. Your task is to find the number of times the subsequence `IMC` appears, with the three letters evenly spaced out. That is, you want to find the number of unique triples $i < j < k$ such that

- $j - i = k - j$,
- The ith letter in the banner is an `I`,
- The jth letter in the banner is an `M`, and
- The kth letter in the banner is a `C`.

Input

The first line of input contains the integer N, the number of characters in the banner.
The second line contains a string of N uppercase letters, describing the banner.

You should read from standard input.

Constraints

For all test cases:

- $1 \leq N \leq 300\,000$.
- Each character of the string is an uppercase letter.

Additionally:

- For Subtask 1 (10% of points), $N \leq 1000$.
- For Subtask 2 (90% of points), there are no additional constraints.

Output

Output a single integer, the number of times the subsequence `IMC` appears, with the three letters evenly spaced out.

You should write to standard output.

Sample Input 1

```
9
IIXMMCIMC
```

Sample Output 1

```
3
```

Explanation 1

The 3 valid subsequences are indicated below.

```
IIXMMCIMC
IIXMMCIMC
IIXMMCIMC
```

Sample Input 2

```
6
KIMCHI
```

Sample Output 2

```
1
```

Scoring

For each subtask (worth 10% and 90% of points, as per the Constraints section), your program will be run on multiple secret test cases one after another, and if it produces the correct output for **all** test cases, it solves that subtask. Your program will receive the points for each subtask it solves. Recall that your final score on the task is the score of your highest scoring submission.