0_Hello World

(30 points)

Time Limit: 1 second
Memory Limit: 256MB

Introduction

YTP Contest has started!

Let's verify everything first.

Is the internet setting correct?

Is the source code submission working well?

Do you use STDOUT output for program solutions?

Everything is ready! Go get 30 points now!! Go! Go! Go!

Statement

Please write a program to output Hello World!

Input Format

This problem requires no input.

Output Format

[A~Z][a~z], space, and common English punctuation.

Constraints

[A~Z][a~z], space, and exclamation mark "!".

Test Cases

Input 1

(no input)

Output 1

```
Hello World!
```

Illustrations

Input 1 has no input, simply output Hello World!

1_I_Hope_You_Can_Be_an_Adult_Someday

(10 points)

Time Limit: 1 second
Memory Limit: 256MB

Statement

"It's raining after all."

The compared child watching the overacasting skies, counting the days to become an adult.

"I hope you can be an adult someday."

You know that becoming an adult isn't something that simply happens when you turn 20, but you can still help him calculate that how many more years it will take for the compared child to become 20, or tell him that he's already 20 and no longer a child.

Input Format

The input contains an integer, the age of the compared child, X.

Output Format

Output an integer, if the compared child is already 20 years old or older, output -1; otherwise, output the number of years it takes to become 20 years old.

Constraints

- $0 \leq X \leq 100$

Test Cases

Input 1

```
19
```

Output 1

```
1
```

Input 2

```
20
```

Output 2

```
-1
```

Input 3

```
30
```

Output 3

```
-1
```

Illustrations

For Input 1, it takes 1 more year for the compared child to become 20.

For Input 2 and 3, the compared child is already 20 years old or older, so output -1.

2_Bomb

(10 points)

Time Limit: 1 second

Memory Limit: 256 MB

Statement

Annie is the president of a hip-hop dance club with 12 members. To help the members quickly bond, she introduced an exciting team-building game in the first class. The rules are as follows:

All members form a circle and are numbered sequentially in a clockwise direction (from number 1 to number 12). The president stands at the center as the dealer. The dealer will tie a card from a deck of playing cards to each person's head in ascending order of their numbers, so no one knows the card tied to their own head. Additionally, the dealer records each person's score, initially set to 1000 points.

At the start of the game, each member takes turns calling out numbers on the playing cards. Number 1 calls out 1, number 2 calls out 2, and so on up to number 12, then number 1 calls out 13, and number 2 starts from 1 again, and so on. If someone calls out a number that matches the card on their own head, they lose and face a punishment designated by the dealer. After the punishment, the dealer ties a new card to the loser's head and deducts the card's value from their score. The next round then begins with the next person in sequence, starting from 1 again, and the game continues for m rounds.

The game is thrilling because the losers do not know how many points they have lost until the scores are revealed at the end of the game. Since there is only one dealer (the president), she worries that calculation errors might be unfair to others, so she asks you to help calculate everyone's total score to ensure accuracy.

Note: Each time a new card is tied, the dealer will draw the top card from the deck.

Input Format

The first line contains 52 integers a_i, separated by spaces, representing the order of the deck of playing cards from top to bottom (each number appears 4 times).

The second line contains an integer m, representing the number of rounds the president wants to perform.

Output Format

Output an integer representing the total score of all members.

Constraints

- $1 \leq a_i \leq 13$
- $1 \leq m \leq 40$

Test Cases

When reading the following test cases, please pay attention to the **Input Format** instructions.

Due to the limited line width on printed pages, the first line of the following test cases **input 1** , **input 2**, and **input 3** have some data displayed on the second line. (It may appear as the second line, but it is actually data from the first line!)

Input 1

```
6 2 9 1 3 9 1 2 5 1 6 13 13 3 9 13 12 4 7 10 11 5 7 8 10 1 9 4 11 4 8 12 12 11 6 7 10 12
11 5 5 6 3 3 13 8 7 8 4 2 10 2
1
```

Output 1

```
11987
```

Input 2

```
6 2 9 1 3 9 1 2 5 1 6 13 13 3 9 13 12 4 7 10 11 5 7 8 10 1 9 4 11 4 8 12 12 11 6 7 10 12
11 5 5 6 3 3 13 8 7 8 4 2 10 2
2
```

Output 2

```
11984
```

Input 3

```
6 2 9 1 3 9 1 2 5 1 6 13 13 3 9 13 12 4 7 10 11 5 7 8 10 1 9 4 11 4 8 12 12 11 6 7 10 12
11 5 5 6 3 3 13 8 7 8 4 2 10 2
40
```

Output 3

```
11694
```

Illustrations

In Example 1, the president sequentially draws 12 cards from the deck and attaches them to the members' heads. At this point, the cards on the members' heads are $[\ 6\ 2\ 9\ 1\ 3\ 9\ 1\ 2\ 5\ 1\ 6\ 13\ ]$. When the game starts, Member 1 calls 1, and Member 2 calls 2. Since the number called matches the card on Member 2's head, Member 2 must accept a penalty. After the penalty, the dealer draws a card (13) from the deck and attaches it to Member 2's head, then deducts 13 points from Member 2's score on the scoreboard. Thus, at the end of the game, the scores are:

- Member 2: 987
- All others: 1000

The total score is 11987.

In Example 2, continuing from the situation in Example 1, since Member 2 lost, the game restarts from Member 3. Member 3 calls 1, Member 4 calls 2, and Member 5 calls 3. Once again, the number called matches the card on Member 5's head, so Member 5 must accept a penalty. After the penalty, the dealer draws a card (3) from the deck and attaches it to Member 5's head, then deducts 3 points from Member 5's score on the scoreboard. Thus, at the end of the game, the scores are:

- Member 2: 987
- Member 5: 997
- All others: 1000

The total score is 11984.

3_Omelet_Loves_Prime_Number

(15 points)

Time Limit: 1 second
Memory Limit: 256 MB

Statement

Omelet has N numbers $a_1, a_2, \ldots, a_N$, all of which are distinct. Since Omelet loves prime numbers, he wants to select some of these numbers such that the sum of any two selected distinct numbers is a prime number. Can you help him calculate the total number of such sequences?

More formally, please calculate how many sequences $b_1, b_2, \ldots, b_K$ satisfy:

- $1 \leq b_1 < b_2 < b_3 < \ldots < b_K \leq N$
- For all $1 \leq i < j \leq K$, $a_{b_i} + a_{b_j}$ is a prime number

Input Format

The first line contains a positive integer N, representing the number of numbers.

The second line contains N positive integers $a_1, a_2, \ldots, a_N$, where a_i represents the i-th number.

Output Format

Output a single integer, representing the number of sequences that meet the criteria. It is guaranteed that within the given constraints, the answer will fit within a 64-bit integer.

Constraints

- $2 \leq N \leq 60$
- $1 \leq a_i \leq 1000$
- $a_i \neq a_j \forall 1 \leq i < j \leq N$

Test Cases

Input 1

```
3
3 1 4
```

Output 1

```
5
```

Input 2

```
5
2 4 6 8 10
```

Output 2

```
5
```

Illustrations

In the first example, there are five possible selections:

- $[3]$
- $[1]$
- $[4]$
- $[3, 4]$
- $[1, 4]$

4_Elevators

(3 points/3 points/ 9 points)

Time Limit: 2 seconds

Memory Limit: 256 MB

Statement

When planning the construction of a new tech building, the design of the elevator system is obviously a key consideration. The elevator system discussed here includes important information such as the floor range and travel time of each elevator, which will directly affect the traffic efficiency inside the building and the comfort of the users.

With a well-designed elevator design system, we can effectively reduce the time it takes for people to reach specific floors from the ground floor of the building. This is crucial for improving the overall operational efficiency of the building and providing a better user experience.

We are currently planning a building with a height of H floors, inside of which there are N elevators. Each elevator has three pieces of information: d_i, u_i, and t_i, which represent that the elevator can move from floor d_i to floor u_i (i.e., the closed interval $[d_i, u_i]$, including the endpoints), and the time it takes to move one floor, t_i seconds. Without considering the time needed for transferring elevators and waiting, what is the shortest time from the ground floor to floor H? If it's impossible to reach floor H from the ground floor, please output -1.

Input Format

The first line of input contains two positive integers H and N, representing the height of the building and the number of elevators.

Following are N lines of input, each containing three positive integers d_i, u_i, and t_i, as described in the problem statement.

Output Format

Output a single positive integer, representing the shortest time needed. If it's impossible to reach floor H from the groud floor, output -1.

Constraints

- $1 \leq H \leq 10^9$.
- $1 \leq N \leq 3 \times 10^5$.
- $1 \leq d_i \leq u_i \leq H$.
- $1 \leq t_i \leq 10^3$.

Subtasks

- Subtask 1 satisfies that $1 \leq H \leq 10^5$, $1 \leq N \leq 10^3$.

- Subtask 2 satisfies that $1 \leq H \leq 3 \times 10^5$, $1 \leq N \leq 3 \times 10^5$.
- Subtask 3 has no additional constraints.

Test Cases

Input 1

```
50 3
20 50 5
35 50 2
1 30 3
```

Output 1

```
142
```

Input 2

```
50 3
20 40 5
45 50 2
1 30 3
```

Output 2

```
-1
```

Input 3

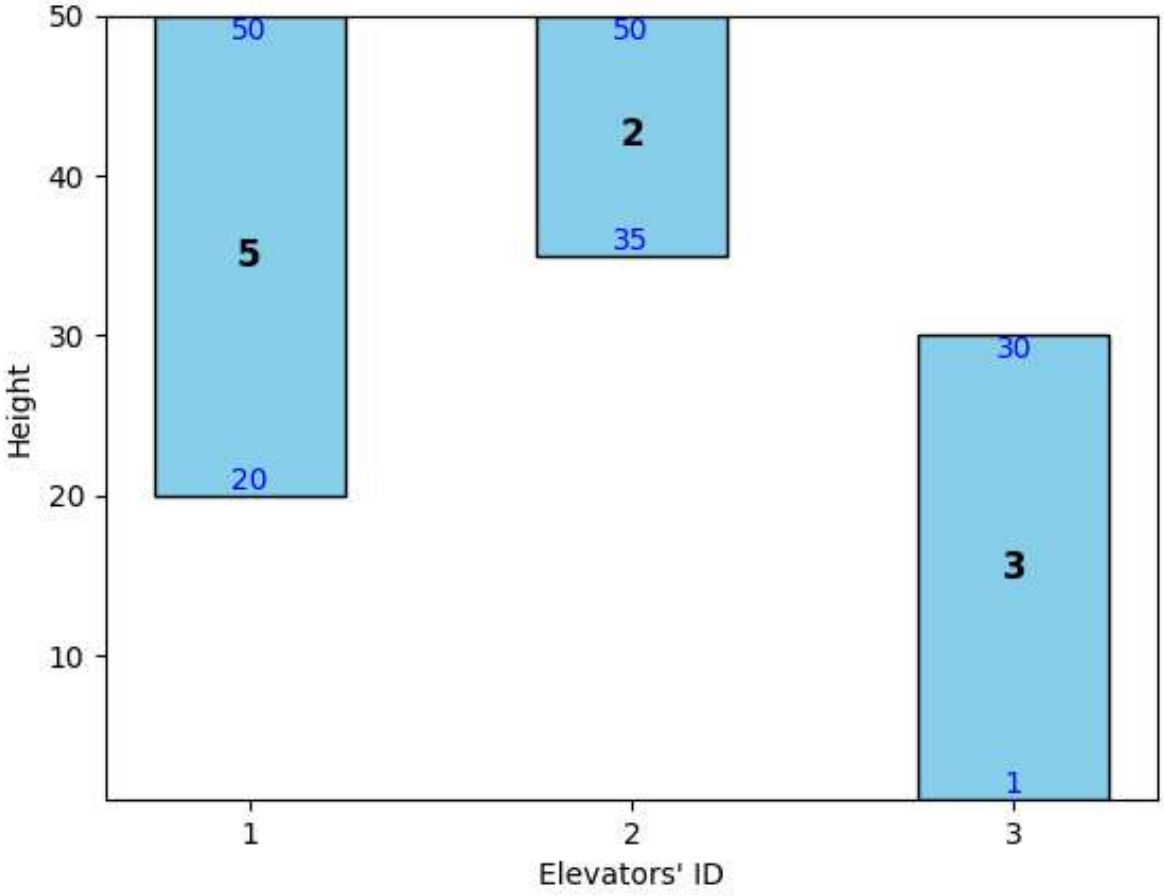
```
50 7
45 50 5
32 48 7
23 40 6
1 28 10
10 35 8
37 50 12
1 13 5
```

Output 3

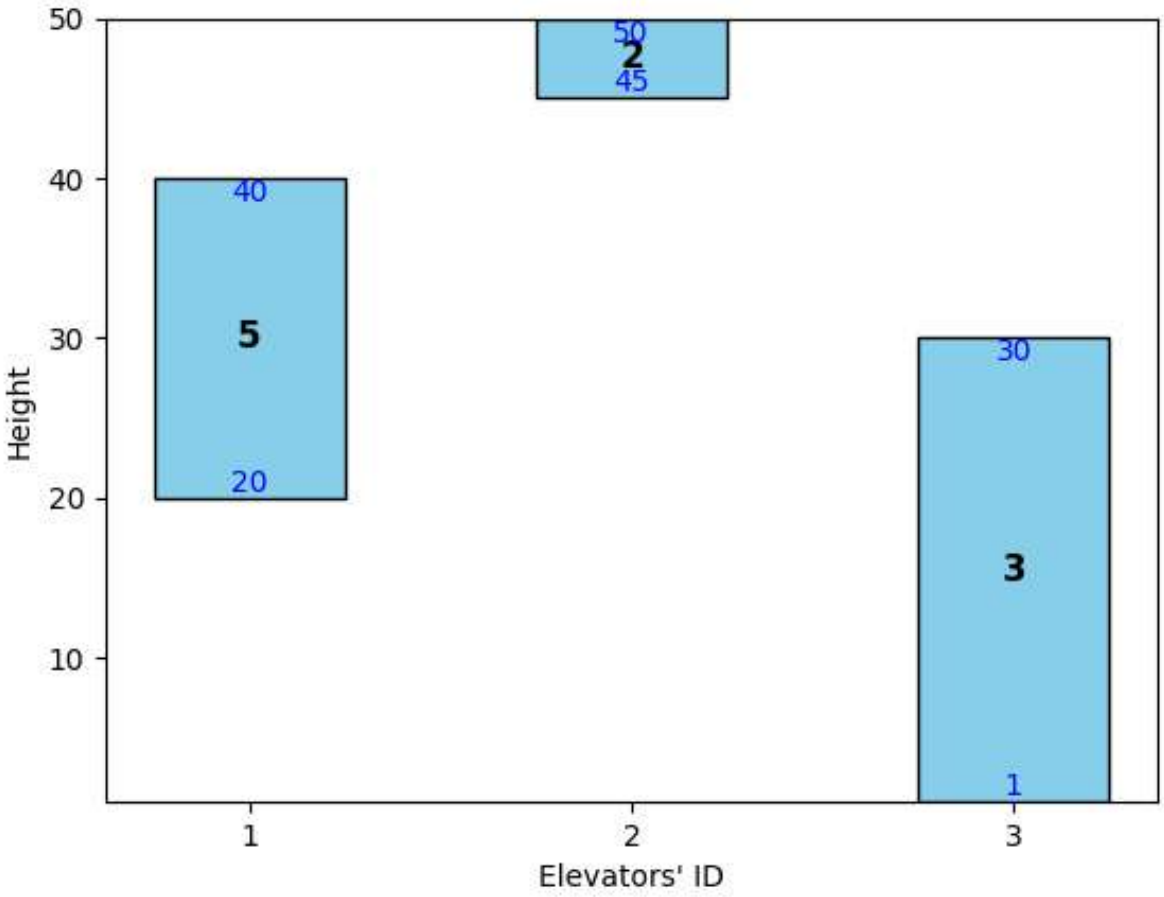
```
302
```

Illustrations

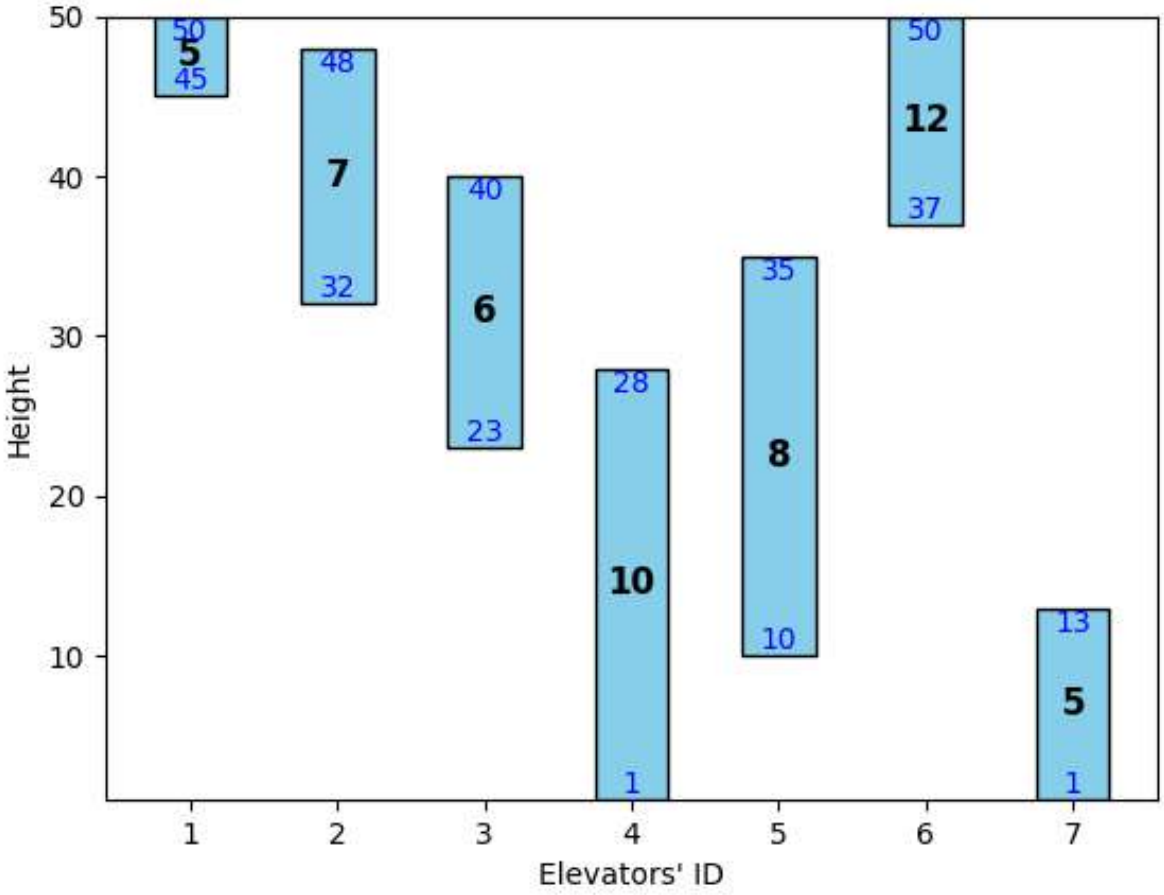
In Example 1, from floors 1 to 30, the third elevator can be used. From 30 to 35, the first elevator can be used. From 35 to 50, the second elevator can be used. The total time spent is $29 \times 3 + 5 \times 5 + 15 \times 2 = 112$.



In Example 2, there is no elevator available for floors 40 to 45, so it's impossible to reach floor H from the ground floor, hence output is -1.



In Example 3, from floors 1 to 13, the seventh elevator can be used. From 13 to 23, the fifth elevator can be used. From 23 to 40, the third elevator can be used. From 40 to 45, the second elevator can be used. From 45 to 50, the first elevator can be used. The total time spent is
$12 \times 5 + 10 \times 8 + 17 \times 6 + 5 \times 7 + 5 \times 5 = 302$.


50
40
30
20
10
Height
50
5
45
48
7
32
40
6
23
28
10
1
35
8
10
50
12
37
13
5
1
1
2
3
4
5
6
7
Elevators' ID

5_Crafting_Axes

(15 points)

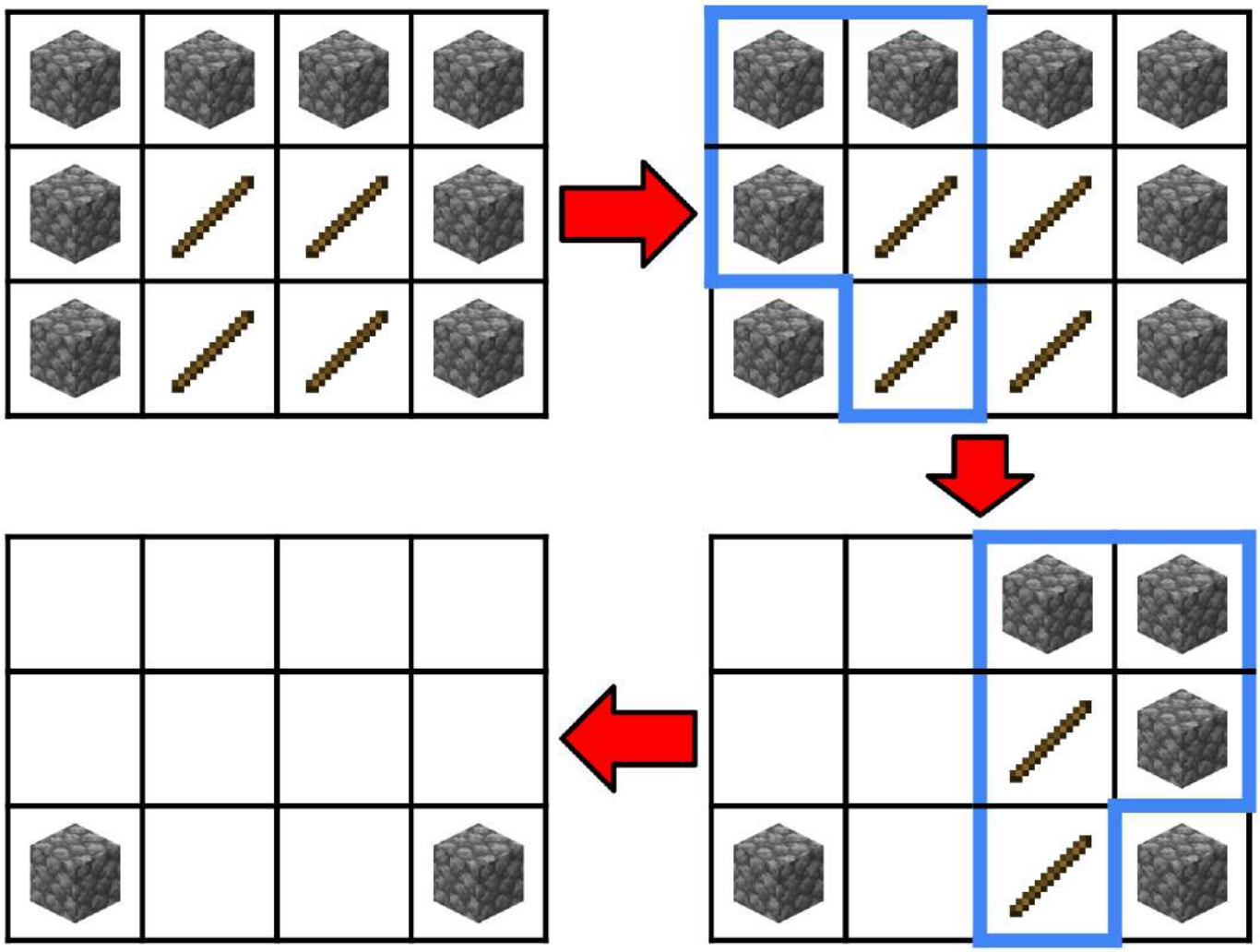
Time Limit: 1 second
Memory Limit: 256 MB

Statement

LittleCube loves to play cube games. He has an $N \times M$ grid, where each cell contains either a stone or a stick. LittleCube wants to craft as many stone axes as possible using the grid. Let (i, j) denote the cell in the i-th row and j-th column. Each time, LittleCube can perform one of the following two operations:

- Select a cell (i, j) satisfying $1 \leq i \leq N-3, 1 \leq j \leq M-2$, such that the cells (i, j), $(i+1, j)$, $(i, j+1)$ contain stones, and the cells $(i+1, j+1)$, $(i+2, j+1)$ contain sticks. LittleCube can use these cells to craft a stone axe, after which these positions will become empty.
- Select a cell (i, j) satisfying $1 \leq i \leq N-3, 1 \leq j \leq M-2$, such that the cells (i, j), $(i, j+1)$, $(i+1, j+1)$ contain stones, and the cells $(i+1, j)$, $(i+2, j)$ contain sticks. LittleCube can use these cells to craft a stone axe, after which these positions will become empty.

For example, the following is an illustration of a valid crafting scenario:



Given the contents of the grid, please help LittleCube calculate the maximum number of stone axes he can craft.

Input Format

The first line contains two positive integers N and M, representing the size of the grid.

The next N lines, each containing a string s_i of length M, where $s_{i,j} = 0$ means the cell (i, j) contains a stone, and $s_{i,j} = 1$ means it contains a stick.

Output Format

Output a single integer, representing the maximum number of stone axes that can be crafted.

Constraints

- $1 \leq N, M \leq 500$
- $|s_i| = M$
- $s_{i,j} \in \{0, 1\}$

Test Cases

Input 1

```
3 4
0000
0110
0110
```

Output 1

```
2
```

Input 2

```
5 5
00000
00000
00000
00000
00000
```

Output 2

```
0
```

Input 3

```
4 4
0000
0100
0110
1110
```

Output 3

```
2
```

Illustrations

The example in the problem statement illustrates the first test case, where the answer is 2.

In the second test case, since there are no sticks, the answer is obviously 0.

6_Bubblelee_Loves_Bubble_Tea

(15 points)

Time Limit: 1 second
Memory Limit: 256 MB

Statement

Little Circle Village is a village situated on a hill, and behind its seemingly ordinary appearance lies an astonishing secret. Beneath the ground of this village, there is a hidden underground lake made entirely of bubble tea! This underground bubble tea lake supplies the entire village with drinking water. According to statistics, the average villager in Little Circle Village consumes 1.5 liters of bubble tea per day. Every morning, the residents bring a 2-liter measuring cylinder to the edge of the underground bubble tea lake and fill a super-sized cup of bubble tea to use as their daily drinking water.

BubbleLee grew up in Little Circle Village, and as a member of the village, he absolutely loves drinking bubble tea! However, due to work obligations, he had to leave Little Circle Village and move to the Big Square Town to make a living. To his surprise, he discovered that there are numerous bubble tea vendors in Big Square Town! Excitedly, he bought N cups of bubble tea from different vendors and returned to his rented place to enjoy them.

However, he realized that the pearl-to-tea ratio in these cups of bubble tea varied significantly. The pearl-to-tea ratio refers to the proportion of pearls to tea in a cup of bubble tea (i.e., $\frac{pearl\ quantity}{tea\ quantity}$). As a bubble tea enthusiast, BubbleLee has his own principle: he cannot accept a situation where, out of the M cups of bubble tea he drinks, the difference in pearl-to-tea ratio between any two cups is greater than K. Now, he wants to maximize the tea quantity he drinks, considering his principle, what is the maximum amount of tea he can consume by selecting some cups out of the N cups of bubble tea?

Input Format

The first line contains three positive integers N, x, and y, which represent the information of N cups of pearl milk tea. x and y represent $K = \frac{x}{y}$.

The following N lines each contain two positive integers b_i and m_i, representing the quantity of pearls and the quantity of tea in the i-th cup of pearl milk tea, respectively.

Output Format

Please output the maximum amount of tea that BubbleLee can drink.

Constraints

- $1 \leq N \leq 2 \times 10^5$
- $1 \leq x, y, b_i, m_i \leq 10^6$

Test Cases

Input 1

```
5 1 4
4 5
5 10
3 3
7 8
7 9
```

Output 1

```
25
```

Input 2

```
10 1 10
200 1500
100 1000
150 2000
70 500
80 700
50 1600
600 2500
202 143
2061 411
2161 411
```

Output 2

```
5700
```

Illustrations

In Example 1, we can choose to drink the 1st, 3rd, 4th, and 5th cups of milk tea, which would result in a total of $5 + 3 + 8 + 9 = 25$ units of milk tea consumed.

In Example 2, we can choose to drink the 1st, 2nd, 3rd, 4th, and 5th cups of milk tea, which would result in a total of $1500 + 1000 + 2000 + 500 + 700 = 5700$ units of milk tea consumed.

7_Traversing_Crossroads

(20 points)
Time Limit: 1 second
Memory Limit: 512MB

Statement

"Oh no! I'm almost late for the contest!" You, a contestant of the annual YTP finals, need to get to the contest hall from the train station as soon as possible.

The area between the station and the contest hall can be represented by N roads from east to west and M roads from north to south. East-west bound roads are numbered from 1 to N from north to south, and north-south bound roads are numbered from 1 to M from west to east. Let (i, j) denote the intersection between the i-th east-west bound road and the j-th north-south bound road. The train station is located at $(1, 1)$, and the contest hall is at (N, M).

It takes one minute to walk between any two adjacent roads with the same direction (that is, one minute to go from (i, j) to either $(i + 1, j)$ or $(i, j + 1)$). The state of the traffic lights in this area can be represented by a string S. If $S[i]$ is `W`, then on the i-th minute, the lights are red on all crossroads along east-west bound roads, and green on others. If $S[i]$ is `H`, then on the i-th minute, the lights are red on all crossroads along north-south bound roads, and green on others. If $S[i]$ is `.`, then all lights are green. You arrived at the station at minute 1.

There are some obstacles amongst these intersections that you cannot pass through. Because you're running low on time, you don't want to waste any time waiting for red lights and taking a detour, so if you are currently at $(i, j) \neq (N, M)$, you need to be at $(i + 1, j)$ or $(i, j + 1)$ after one minute. You start to wonder, how many possible paths could you take?

Given a map of the intersections and state of traffic lights, write a program to determine the number of shortest paths you could take without waiting for red lights, modulo $10^9 + 7$.

Note: The modulo operation for integers a and $b > 0$ is defined as the smallest non-negative integer r such that $a = bq + r$. In other words, the remainder of a divided by b.

Input Format

The first line of the input contains two integers N, M

The second line contains a string S of length $N + M - 2$, denoting the state of the traffic lights. It can be shown that the shortest path without waiting for any red lights takes exactly $N + M - 2$ minutes.

The next N lines each contain M integers. Let $a_{i,j}$ be the j-th integer on the i-th line. If $a_{i,j} = 0$, the intersection (i, j) can be traversed. If $a_{i,j} = 1$, there is an obstacle on intersection (i, j).

Output Format

Output one integer, the number of shortest paths you could take without waiting for red lights, modulo $10^9 + 7$.

Constraints

- $1 \leq N, M \leq 1000$
- $S[i] \in \{H, W, .\}, \forall 1 \leq i \leq N + M - 2$
- $a_{i,j} \in \{0, 1\}, a_{1,1} = 0, a_{N,M} = 0$

Subtasks

There is only one subtask to this problem.

Test Cases

Input 1

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

Output 1

```
4
```

Input 2

```
4 4
.H.W..
0 0 0 0
0 0 1 0
0 0 1 0
1 0 0 0
```

Output 2

```
2
```

Illustrations

In sample test 2, the 2 possible paths are as follows (the crossroads traversed are denoted with an x).

```
x 0 0 0
x x 1 0
0 x 1 0
1 x x x

x x x x
0 0 1 x
0 0 1 x
1 0 0 x
```

8_Farm

(1 points/4 points/15 points)

Time Limit: 1 second

Memory Limit: 256 MB

Statement

Once upon a time, there was a vast farm stretching for several kilometers. This entire farm could be represented by the x-axis on a number line. On this farm, several cows were tethered to stakes on the x-axis with ropes of varying lengths, restricting their movement to $[x - r, x + r]$, where x represents the position of the stake and r represents the length of the rope. Each cow also had a specific hunger value t, indicating that they would consume t liters of food each time they ate.

Due to the farmer's extreme laziness, he decided to set up some automatic feeders to reduce the daily fatigue of feeding the cows. To ensure the cows eat in an orderly fashion, the farmer would first have the cows line up according to the x-coordinates of their stakes. This means the cows with stakes at smaller x-coordinates would be at the front of the line. The farmer would then activate the automatic feeders, allowing the cows to eat one after another. Since the cows positioned further to the right would receive food later, their hunger value would increase. Each cow's hunger value would become its initial hunger value plus the sum of the initial hunger values of all the cows ahead of it in the line.

Because the farmer has to manage many cows at once, he is unsure where to place the automatic feeders most effectively. Therefore, he has listed some predetermined locations and would like you to help him calculate the minimum amount of food in liters needed if an automatic feeder is placed at each of those locations to satisfy all the cows that can reach the feeder.

Input Format

The first line contains two positive integers, N and M, representing the number of cows and the number of automatic feeding machines, respectively.

Following that, there will be N lines, each containing three integers, x_i, r_i, t_i, which respectively denote the x-coordinate, rope length, and the initial hunger value of the i-th cow.

The last line contains M integers $a_1, a_2, \ldots, a_M$, representing the x-coordinates of the i-th predetermined location for the automatic feeders.

Output Format

Output a single line containing M integers separated by spaces, representing the minimum amount of food needed for each automatic feeding machine to satisfy all cows that can reach it.

Constraints

- $1 \leq N, M \leq 2 \times 10^5$.
- $1 \leq x_i, r_i, t_i, a_i \leq 10^8$.
- $\forall i \neq j, x_i \neq x_j, a_i \neq a_j$.

Subtasks

- Subtask 1 satisfies that $M \leq 10$.
- Subtask 2 satisfies that $r_i \leq 50, x_i, a_i \leq 10^6$.
- Subtask 3 has no additional restrictions.

Test Cases

Input 1

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

Output 1

```
10
```

Input 2

```
3 1
2 3 5
3 3 2
7 3 6
5
```

Output 2

```
25
```

Input 3

```
5 4
1 100 7
5 8 5
10 1 4
21 3 2
53 10 6
11 19 63 64
```

Output 3

```
35 16 20 7
```

Illustrations

In Example 1, there are two cows located at $x = 2$ and $x = 6$, both with a rope length of 2. This means the first cow's range of movement is $[0, 4]$, and the second cow's range is $[4, 8]$. Therefore, an automatic feeder placed at $x = 4$ can feed both cows. After lining up the cows according to the x-coordinates of their stakes, the first cow's hunger value is 3 liters, and the second cow's hunger value is $3 + 4 = 7$ liters. Thus, the feeder must prepare at least $3 + 7 = 10$ liters of food.

In Example 2, all three cows can reach the automatic feeder. After lining up the cows according to the x-coordinates of their stakes, the first cow's hunger value is 5 liters, the second cow's hunger value is $5 + 2 = 7$ liters, and the third cow's hunger value is $5 + 2 + 6 = 13$ liters. Therefore, the feeder must prepare at least $5 + 7 + 13 = 25$ liters of food.

9_No_Coin

(4 points/16 points)

Time Limit: 2 seconds

Memory Limit: 1024 MB

Statement

Little Y is an ordinary citizen living in the country YTP. He works with diligence every day, leading a mundane and regular life. However, his life was about to undergo a significant turning point.

On the evening of day 0, as he was walking home from work, he suddenly spotted a shiny stone on the ground. Driven by curiosity, he bent down and picked it up, only to discover that underneath the stone was a lottery ticket.

Little Y thought to himself that perhaps this was his lucky day, so he hurried home to check the lottery numbers. As he compared them with the lottery results broadcasted on TV, he couldn't believe his eyes, he had indeed won the jackpot!

Filled with excitement, Little Y stayed up all night, repeatedly looking at the lottery ticket in his hand, making sure he wasn't dreaming. That night, his life changed completely.

After claiming the prize, Little Y instantly became a millionaire, with a total fortune of $10^{1000000}$ YTP dollar. However, thrifty Little Y still maintained his good habit of planning his expenses carefully. He plans to spend a_i YTP dollar each day for the next n days. However, receiving so much money also caused trouble for Little Y, one of the biggest concerns is the problem of small change.

In the currency of the country YTP, there are different denominations, among which only 1 YTP dollar is in coin form. However, due to the particularly cheap price of the metal "osmium" in YTP country, all coins are made of highly dense osmium. Therefore, Little Y dislikes carrying the heavy 1 YTP dollar coin.

Knowing Little Y's dilemma, Little T decided to design a digital payment system to make it easier for Little Y to pay. However, since Little T has never participated in the YTP program, the system he designed can only be used for up to k days within n days period. Therefore, Little Y turned to Little P, who has participated in the YTP program, to design a plan to decide whether to use cash or digital payment for each day, so that **the total amount of remaining coins after each day's payment is minimized**.

As a teammate of Little P, please design a program with him to calculate **the minimum total amount of remaining coins each day after payment**, and determine on which days digital payment should be used to achieve the minimum total.

Since Little Y hates coins so much, he always adopts the method of minimizing the number of 1 YTP dollar coins he owns when paying, which means that at any time, Little Y's wallet will never have more than five 1 YTP dollar coins.

Input Format

The input consists of two lines.

The first line consists of two integers n, k, respectively representing the total amount of days and the maximum number of days that Little Y can use digital payment.

The second line consists of n non-negative integers $a_1, a_2, \ldots, a_n$, representing the money that Little Y plans to spend for the next n days.

Output Format

The output consists of two lines.

The first line consists of two non-negative integers $total\_cnt, days$, respectively representing **the minimum total amount of remaining coins each day after payment**, and the total amount of days that Little Y should use digital payment.

The second line consists of $days$ positive integers $ans_1, \ldots, ans_{days}$, respectively representing the index of days that Little Y should use digital payment.

The output's $days$ should satisfy $0 \leq days \leq k$, and ans should be a strictly increasing sequence.

If there are plenty of possible plans to form **the minimum total amount of remaining coins each day after payment**, please output one of any possible plans that have the smallest $days$.

Constraints

- $1 \leq k \leq n \leq 1000$
- $0 \leq a_i \leq 10^8$

Subtasks

- Subtask 1 satisfies that $1 \leq k \leq n \leq 20$.
- Subtask 2 has no additional constraints.

Test Cases

Input 1

```
5 2
1 2 3 4 5
```

Output 1

```
3 2
1 4
```

Input 2

```
9 1
4 2 3 3 5 5 5 5 5
```

Output 2

```
5 1
2
```

Input 3

```
9 3
4 2 3 3 5 15 25 35 45
```

Output 3

```
3 2
1 4
```

Illustrations

In Example 1, Little Y can use digital payment on day $1, 4$, the remaining coins from day 1 to day n are $0, 3, 0, 0, 0$, thus **the minimum total amount of remaining coins each day after payment** is 3.

In Example 2, Little Y can use digital payment on day 2, and the remaining coins from day 1 to day n are $1, 1, 3, 0, 0, 0, 0, 0, 0$, thus **the minimum total amount of remaining coins each day after payment** is 5.

In Example 3, Little Y can use digital payment on day $1, 4$, and the remaining coins from day 1 to day n are $0, 3, 0, 0, 0, 0, 0, 0, 0$, thus **the minimum total amount of remaining coins each day after payment** is 3. Notice that in this example, less than k times of digital payment are used.

10_Letter_Recognition

(6 points /6 points /13 points)

Time Limit: 1 second
Memory Limit: 256MB

Statement

The Yummy Tummy Picnic, or YTP for short, is about to start!

To advertise the activity, Yammy and Pammy prepared a lot of posters with the letters `Y`, `T`, and `P` printed on them. Each poster contains one of the letter. They were about to give these posters to Tammy to paste them somewhere. They planned to give these posters to Tammy to put them up. However, the moment they stepped outside, the strong wind blew, swept back their hair, startled the fighting pigeons, and scattered the posters all over the ground! To sort the posters with the three different letters, they need you to write a program to quickly recognize the letter on the posters.

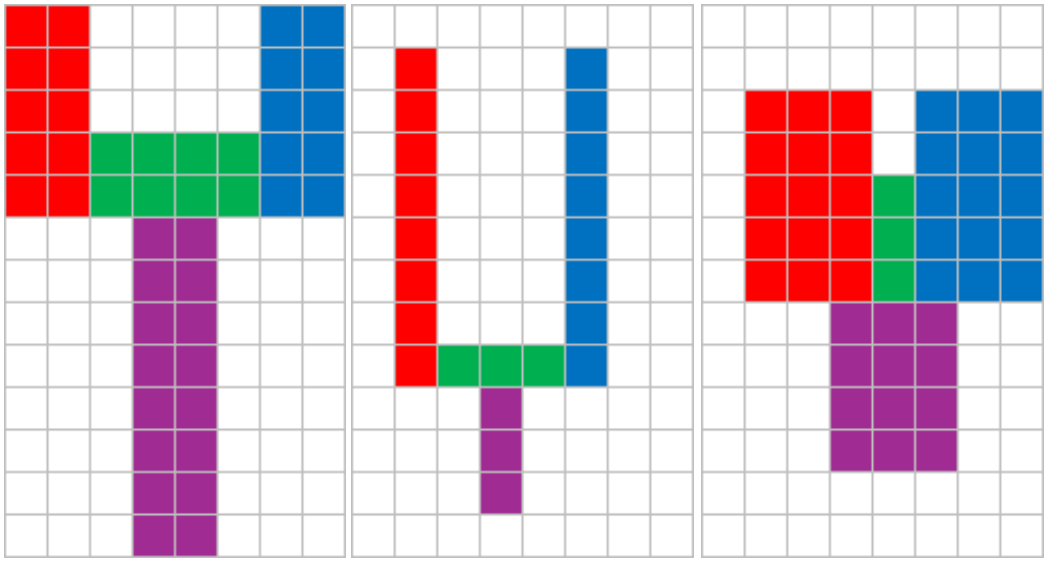
Luckily, the fonts on these posters are all the same, so it won't be a difficult job for you to recognize the letters. Each poster can be considered as an $N \times M$ grid, where each row is numbered from 1 to N from top to bottom, and each column is numbered from 1 to M from left to right. The cell at row i and column j is denoted as (i, j). The color of each cell can be either black or white.

Next, we will define what the letters `Y`, `T`, and `P` looks like. In the below sections, the rectangles we mention are a set consists of cells. For rectangle X, its top-left corner cell is denoted as (u_X, l_X), and its bottom-right corner cell is denoted as (d_X, r_X), and X contains all cells (i, j) such that $u_X \leq i \leq d_X$ and $l_X \leq j \leq r_X$.

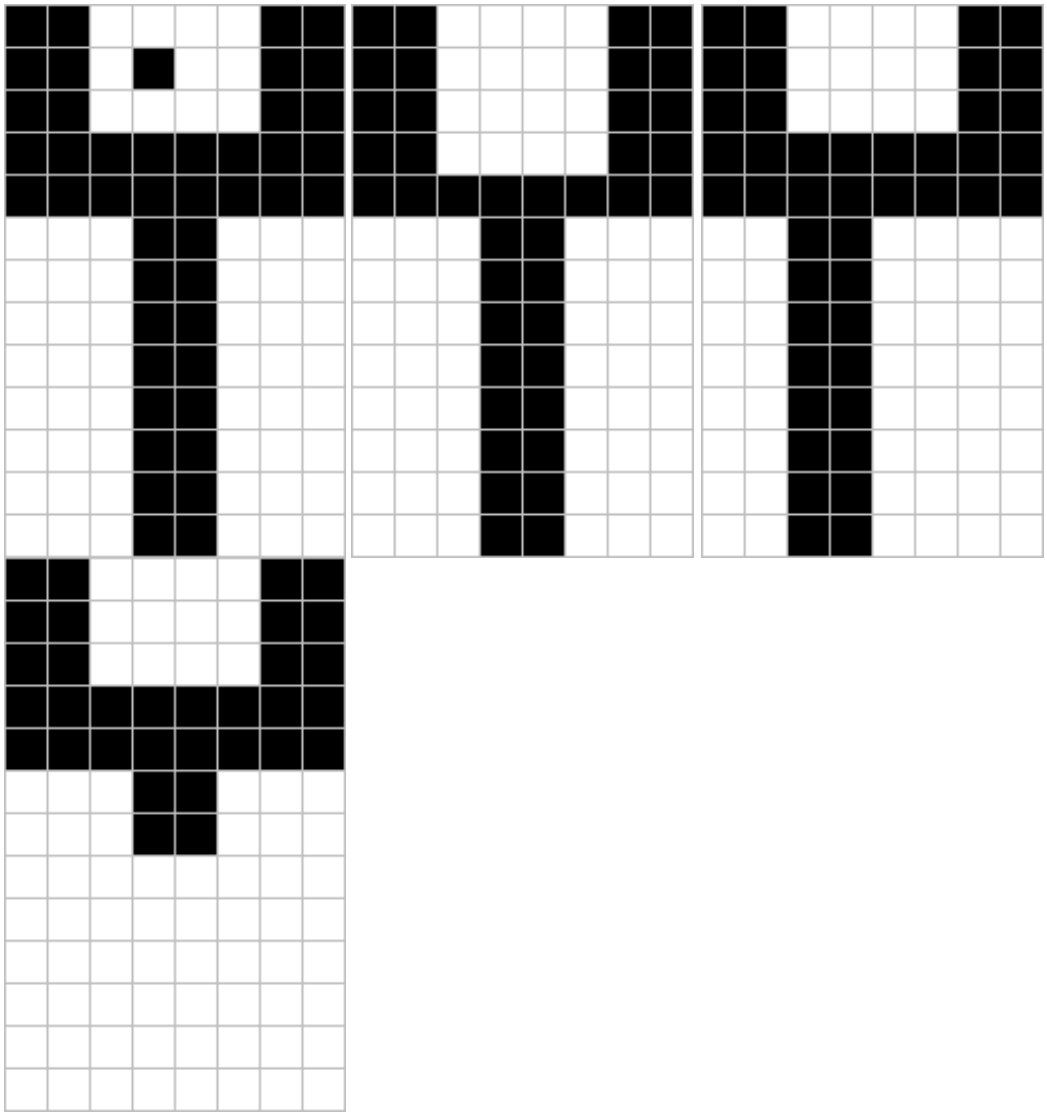
A letter `Y` is divided into four rectangle regions A, B, C, D. A poster must contain such rectangles A, B, C, D satisfying all the following conditions to be considered a poster printed with the letter `Y`.

- $d_A - u_A > r_A - l_A \geq 0$.
- $d_B = d_A$ 且 $d_B - u_B = r_A - l_A$ 且 $l_B = r_A + 1$ 且 $r_B \geq l_B$.
- $d_C = d_A$ 且 $u_C = u_A$ 且 $l_C = r_B + 1$ 且 $r_C - l_C = r_A - l_A$.
- $u_D = d_A + 1$ 且 $r_B - r_D = l_D - l_B$ 且 $r_D - l_D = r_A - l_A$ 且 $d_D - u_D > r_D - l_D$.
- For all cells within any of the rectangles A, B, C, D, their color should be black. For all cells not within rectangles A, B, C, D, their color should be white.

For example, here are some posters printed with the letter `Y` that satisfy the conditions. The black cells are colored to different colors to distinguish the rectangle regions: the red color represents rectangle A, green represents rectangle B, blue represents rectangle C, and purple represents rectangle D.



And below are some posters that do not satisfy all the conditions.

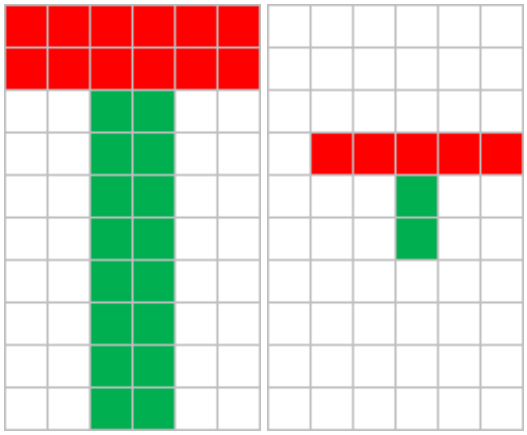


A letter T is divided into two rectangle regions E, F. A poster must contain such rectangles E, F satisfying all the following conditions to be considered a poster printed with the letter T.

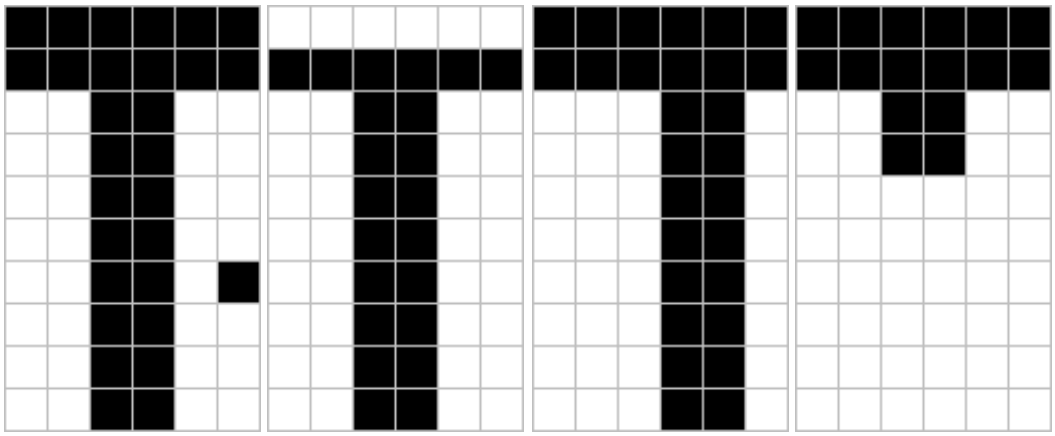
- $r_E - l_E > d_E - u_E \geq 0$。
- $u_F = d_E + 1$ 且 $r_E - r_F = l_F - l_E$ 且 $r_F - l_F = d_E - u_E$ 且 $d_F - u_F > r_F - l_F$。

- For all cells within any of the rectangles E, F, their color should be black. For all cells not within rectangles E, F, their color should be white.

For example, here are some posters printed with the letter T that satisfy the conditions. The black cells are colored to different colors to distinguish the rectangle regions: the red color represents rectangle E, and green represents rectangle F.



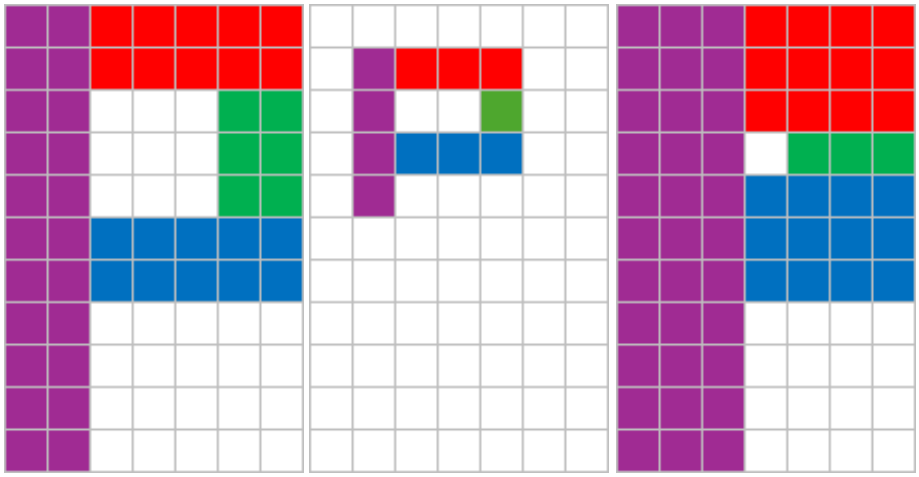
And below are some posters that do not satisfy all the conditions.



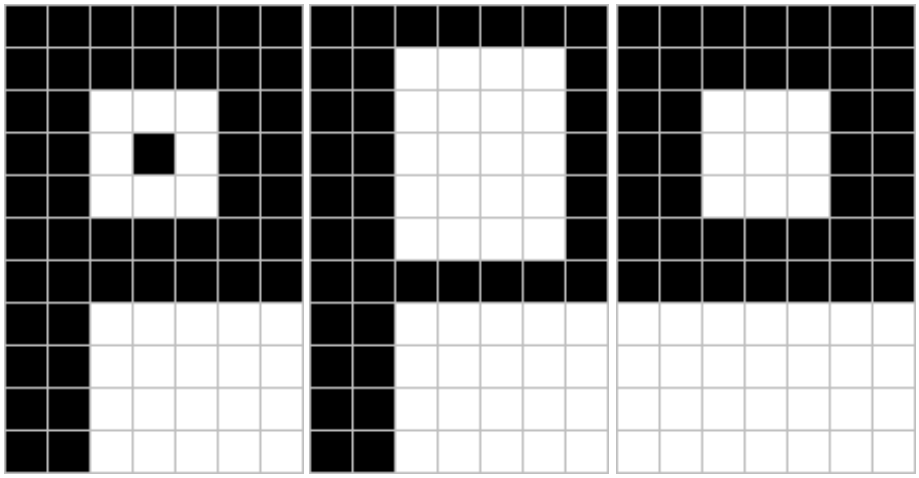
A letter P is divided into four rectangle regions G, H, I, J. A poster must contain such rectangles G, H, I, J satisfying all the following conditions to be considered a poster printed with the letter P.

- $r_G - l_G > d_G - u_G \geq 0$。
- $u_H = d_G + 1$ 且 $d_H \geq u_H$ 且 $r_H = r_G$ 且 $r_H - l_H = d_G - u_G$。
- $u_I = d_H + 1$ 且 $d_I - u_I = d_G - u_G$ 且 $r_I = r_G$ 且 $l_I = l_G$。
- $u_J = u_G$ 且 $d_J > d_I$ 且 $r_J = l_G - 1$ 且 $r_J - l_J = d_G - u_G$。
- For all cells within any of the rectangles G, H, I, J, their color should be black. For all cells not within rectangles G, H, I, J, their color should be white.

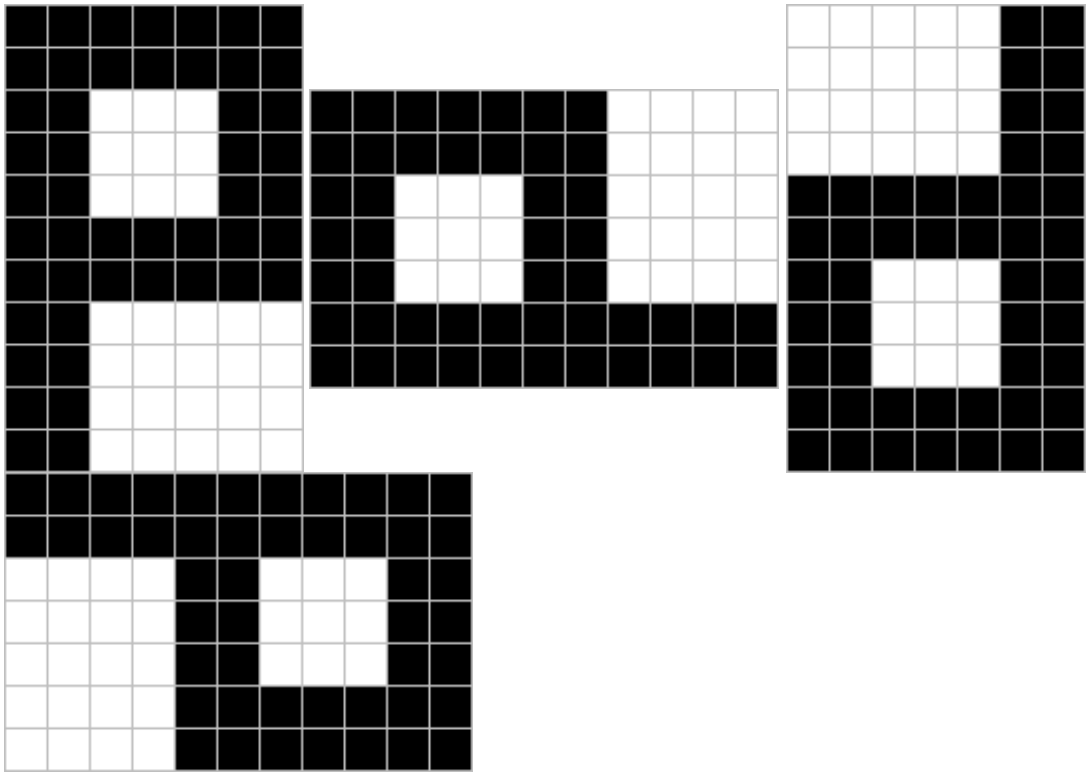
For example, here are some posters printed with the letter P that satisfy the conditions. The black cells are colored to different colors to distinguish the rectangle regions: the red color represents rectangle G, green represents rectangle H, blue represents rectangle I, and purple represents rectangle J.



And below are some posters that do not satisfy all the conditions.



Since the posters were scattered by the wind, they may have been rotated by 0, 90, 180, or 270 degrees after being retrieved. For each poster, you should check all four angles. If any of the rotated angles satisfies the conditions for a letter, the poster will be considered printed with that letter. For example, below are the posters with the letter P rotated by 0, 90, 180, and 270 degrees, respectively.



Now it's time to write a program to recognize the letters on the posters! Please help Yammy, Tammy, and Pammy so that YTP can start without a hitch!

Input Format

The first line of the input contains two space-seperated positive integers N and M, denoting the size of the poster.

In the next N lines of input, each line contains a string of length M. The j-th character on the i-th line shows the color of the cell (i, j). The character `#` means that the cell is black, and the character `.` means that the cell is white.

Output Format

Output a single letter. If the poster is printed with the letter `Y`, `T`, or `P`, output the corresponding uppercase letter. If no letters can be recognized on the poster, output the uppercase letter `X`.

Constraints

- $1 \leq N, M \leq 50$
- The strings contain only characters `#` and `.`

Subtasks

- Subtask 1 satisfies that the input will only contain the poster with letter `T` or `P` printed on it, and every poster of the input is printed with a letter on it.
- Subtask 2 satisfied that every poster of the input is printed with a letter on it.
- Subtask 3 has no additional constraints.

Test Cases

Input 1

```
13 8
##....##
##....##
##....##
########
########
...##...
...##...
...##...
...##...
...##...
...##...
...##...
...##...
```

Output 1

```
Y
```

Input 2

```
10 6
......
......
......
.#####
...#..
...#..
......
......
......
......
```

Output 2

```
T
```

Input 3

```
7 11
#######....
#######....
##...##....
##...##....
##...##....
###########
###########
```

Output 3

```
P
```

Input 4

```
11 7
##.....
##.....
##.....
##.....
#######
#######
##...##
##...##
##...##
#######
#######
```

Output 4

```
X
```

Input 5

```
14 26
..........................
..........##.........##...
.##....##.######..##.###..
.##....##.######..##..###.
.##....##.##......##...##.
.##....##.##...##.##...##.
.###..###.##..###.##..###.
..#######.######..######..
...###.##..####...####....
.......##.........##......
......##..........##......
....####..........##......
....###............##.....
..........................
```

Output 5

```
X
```

Illustrations

Input 1 is the first poster of the letter Y in the problem statement.

Input 2 is the second poster of the letter T in the problem statement.

Input 3 is the poster with letter P rotated by 90 degrees.

No letters can be recognized on the poster of Input 4 and 5, so output X.

11_Fruit_Advantage

(6 points / 19 points)

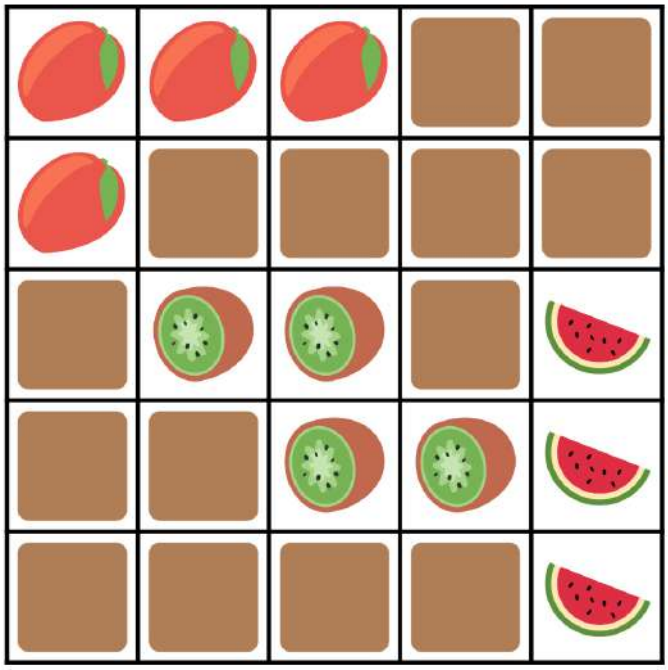
Time Limit: 2 seconds
Memory Limit: 256 MB

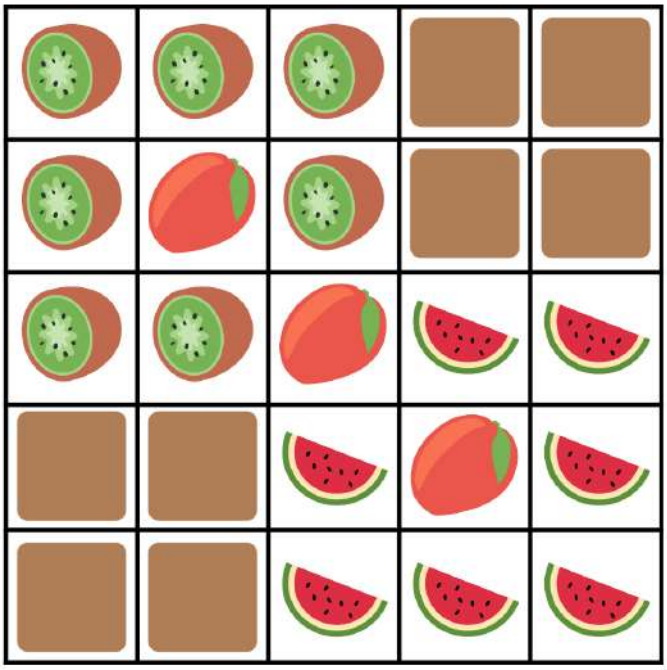
Statement

Wiwi has a field that can be represented by an $N \times M$ grid, where each cell can be used to plant a type of fruit. Recently, Wiwi planted K types of fruits in this field. For easier care, each type of fruit is planted in a connected block. Two cells are considered adjacent if they share an edge. We define the i-th type of fruit to form a connected block if for any two cells s and t planted with the i-th type of fruit, there exists a sequence of cells $c_1, c_2, \ldots, c_k$ planted with the i-th type of fruit such that:

- $c_1 = s$
- $c_k = t$
- For all $1 \leq j < k$, c_j and c_{j+1} are adjacent

For example, in the diagrams below, the brown areas represent empty fields. The left diagram shows a valid planting method, while the right diagram is invalid because the mangoes do not form a connected block.





Now that the fruits are ripe, to make everyone realize the advantages of each type of fruit, Wiwi wants to harvest at least one of each type of fruit. Wiwi will use robots to harvest the fruits, and each robot has two harvesting modes, one of which it can execute:

- Choose a column and harvest all the fruits in that column.
- Choose a row and harvest all the fruits in that row.

Since robots are expensive, Wiwi wants to achieve her goal with the minimum number of robots. Can you help her?

Input Format

The first line contains three positive integers N, M, and K, representing the size of the grid and the number of types of fruits.

The next N lines each contain M integers $a_{i,1}, a_{i,2}, \ldots, a_{i,M}$, where $a_{i,j}$ represents the type of fruit planted in the cell at the i-th row and j-th column. If $a_{i,j} = 0$, it means the cell is empty. Otherwise, it means the cell contains the fruit of type $a_{i,j}$.

Output Format

Output a single integer, representing the minimum number of robots required.

Constraints

- $1 \leq NM \leq 300$
- $1 \leq K \leq NM$
- $0 \leq a_{i,j} \leq K$
- For all $1 \leq x \leq K$, there is at least one $a_{i,j} = x$.

Subtasks

- Subtask 1: Satisfies $1 \leq N, M \leq 7$ (6 points)
- Subtask 2: No additional constraints (19 points)

Test Cases

Input 1

```
5 5 3
1 1 1 0 0
1 0 0 0 0
0 2 2 0 3
0 0 2 2 3
0 0 0 0 3
```

Output 1

```
2
```

Input 2

```
3 4 4
0 1 0 4
2 1 3 4
2 0 3 0
```

Output 2

```
1
```

Input 3

```
3 3 5
1 2 3
4 0 0
5 0 0
```

Output 3

```
2
```

Input 4

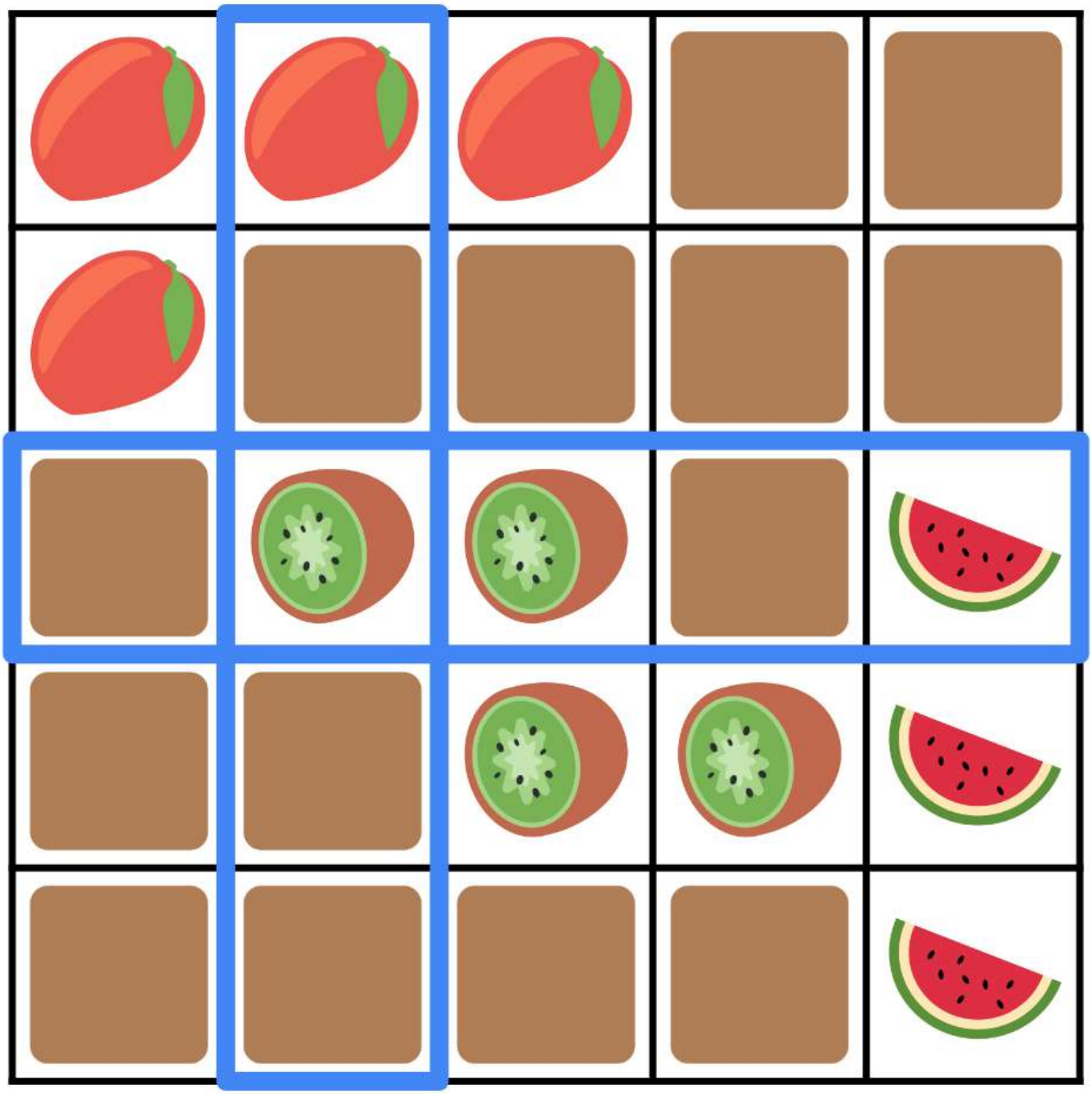
```
4 5 7
0 6 7 3 5
0 0 0 3 5
4 0 0 3 0
1 0 0 2 0
```

Output 4

```
3
```

Illustrations

In the first example, we can use two robots to harvest the fruits within the blue frames to achieve the goal. This is one of the optimal solutions.



12_Staircase

(3 points/8 points/7 points/7 points)

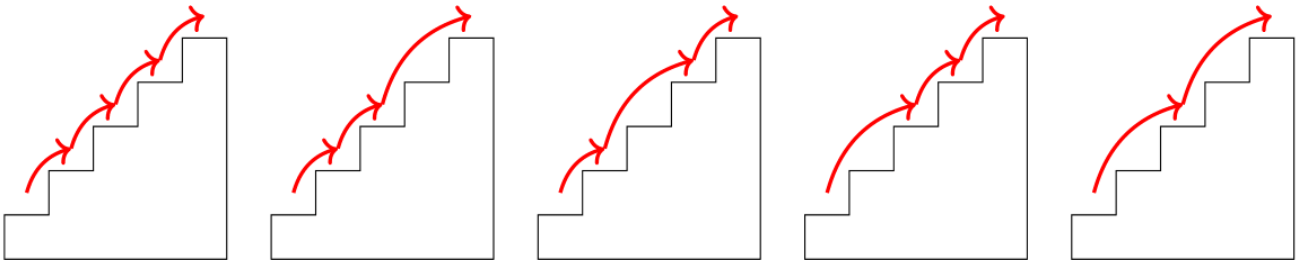
Time Limit: 5 seconds

Memory Limit: 1024 MB

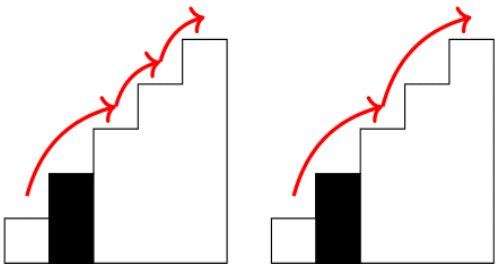
Statement

"Once upon a time, there was a staircase. With each step, you could move up one or two steps. How many ways are there to go from the first step to the x-th step?"

Two ways of stair climbing are considered different if the sets of steps stepped on are different. Smart as you are, you drew every possible way of stair climbing from the first step to the fifth step, and quickly discovered a pattern.



You decided to try climbing the staircase in person to verify your solutions. However, the staircase broke after being stepped on a few times! Broken steps cannot be stepped on before getting fixed, resulting in fewer ways to climb the stairs. For example, if the second step is broken, there will be only two valid ways left to go from the first step to the fifth step.



Now, given records of the staircase getting broken or fixed, can you calculate the number of ways to get from the first step to a certain step?

Input Format

The first line contains two integers N, Q, representing the maximum number of steps and the number of upcomming events.

Each of the following Q lines represents an event:

- $1\ x$: The x-th step gets fixed if it was broken; otherwise, it breaks.
- $2\ x$: We want to know the number of ways to go from the first step to the x-th step without stepping on broken steps, if we move up no more than two steps each time.

Output Format

For each of the events of type 2, output an integer representing the answer modulo 998244353.

Constraints

- $N \leq 10^{18}$
- $Q \leq 5 \times 10^5$
- $1 \leq x \leq N$

Subtasks

- Subtask 1 satisfies that $N, Q \leq 5000$
- Subtask 2 satisfies that all events are of type 2. Stairs never break.
- Subtask 3 satisfies that at any moment, at most 10 steps are broken.
- Subtask 4 has no additional constraints.

Test Cases

Input 1

```
10 6
2 10
1 3
1 8
2 10
1 3
2 10
```

Output 1

```
55
3
13
```

Input 2

```
15 8
2 15
1 1
2 15
1 1
1 5
2 15
1 6
2 15
```

Output 2

```
610
0
165
0
```

Input 3

```
10 10
2 1
2 2
2 3
2 4
2 5
2 6
2 7
2 8
2 9
2 10
```

Output 3

```
1
1
2
3
5
8
13
21
34
55
```

Input 4

```
20 20
2 5
2 18
2 12
2 2
2 7
1 11
1 11
1 2
2 9
1 2
1 6
1 6
1 9
```

```
1 9
2 12
2 9
1 7
1 4
1 10
2 18
```

Output 4

```
5
2584
144
1
13
13
144
34
42
```

Illustrations

In Input 2, if the first step is broken, we cannot even take the first step, so there is no way we can go anywhere.

13_Logic_Circuit

(7 points /9 points /9 points)

Time Limit: 1 second
Memory Limit: 256MB

Statement

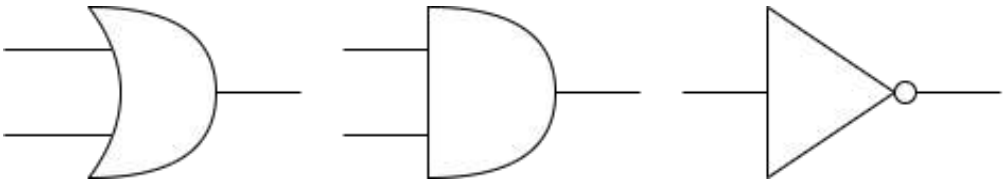
Digital Systems Design and Laboratory is a common course for CS students in many universities. Boolean algebra and digital circuit design are some of the most fundamental topics covered in the course. In this problem, we will write a program that converts a logical expression into a logic circuit diagram. Not familiar with Boolean algebra yet? No worries, let's start with a simple tutorial.

In this problem, the output of the logical expression will always be the uppercase letter `F`, and the remaining 25 uppercase letters will be used as the names of other input variables. Just like in programming languages, the three main operators in Boolean algebra are `OR`, `AND`, and `NOT`. We will introduce them one by one in the following sections.

In Boolean algebra, the `OR` operation is represented similarly to addition in mathematics. For example, if we perform an `OR` operation on variables A and B, we would write it as `A+B`. The `AND` operation is represented similarly to multiplication in mathematics, but we omit the multiplication symbol and simply place the two variables side by side. For example, if we perform an `AND` operation on variables A and B, we would write it as `AB`. The `NOT` operation is represented by adding a single quote after the variable or expression to be negated. For example, if we perform a `NOT` operation on variable `A`, we would write it as `A'`.

The precedence of these three operators is also the same as in most programming languages. First, the contents of the parentheses are evaluated, then the `NOT` operations, followed by the `AND` operations, and finally the `OR` operations.

Next, we will explain how to convert a logical expression into a logic circuit diagram. First, let's introduce the logic gates of these operations. From left to right in the diagram below are the `OR`, `AND`, and `NOT` logic gates. The line segments on the left side are inputs to the logic gates, and the line segments on the right side represents the output of the logic gates. The `OR` and `AND` gates can have two or more inputs, while the `NOT` gate has only one input. All logic gates have exactly one output.



To define the input format precisely, any input to the problem is given by an `<input>` non-terminal of the following BNF (Backus Normal Form) grammar:

```
<input> ::= “F=” <disj>
<disj>  ::= <conj> | <disj> “+” <conj>
<conj>  ::= <atom> | <conj> <atom>
<atom>  ::= <var> | <var> “'” | “(” <disj> “)” | “(” <disj> “)'”
<var>   ::= “A” | “B” | “C” | “D” | “E” | “G” | “H” | “I” | ... | “Z”
```

Below are the explanation of the BNF grammar.

- `<input>`: The input string of the problem. `F` is the output of the logical expression, and `<disj>` is the main logical expression.
- `<disj>`: Consists of a series of `<conj>` seperated by `+` characters, indicating **OR** operations between the logical expressions.
- `<conj>`: Consists of a series of `<atom>` without any characters separating them, indicating **AND** operations between the logical expressions.
- `<atom>`: Can be either a `<var>` or a `<disj>` enclosed in parentheses, and may include a single quote denoting a **NOT** operation on it.
- `<var>`: The variable names of the inputs of the logical expression. Notice that `F` is not in the list.

And below are the rules to convert each non-terminal symbol into the logic circuit diagram.

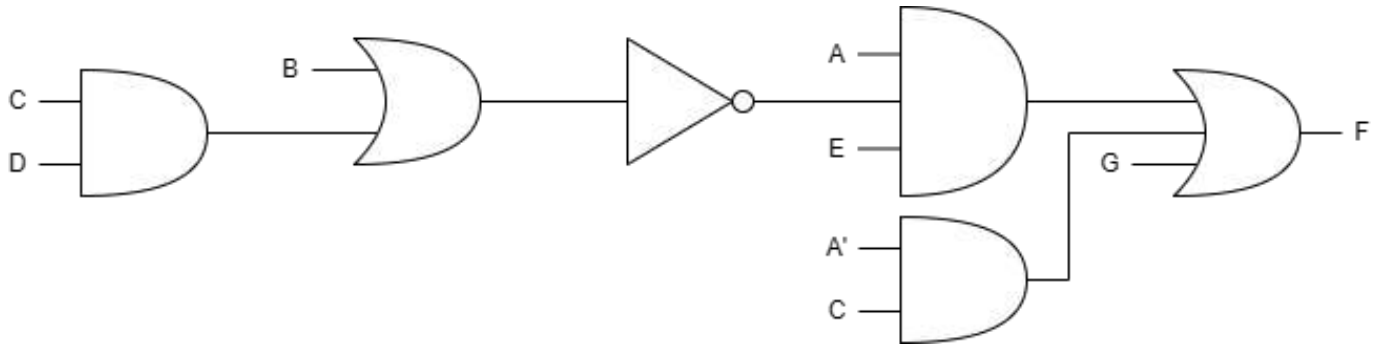
- The `<disj>` in `<input>` is the main circuit, outputting the variable `F`.
- Seperate each `<disj>` into a series of `<conj>`.
  - If `<disj>` consists of only one `<conj>`, treat it as a `<conj>`.
  - Otherwise, connect the outputs of each `<conj>` to the inputs of an **OR** gate, one by one, from top to bottom.
- Seperate each `<conj>` into a series of `<atom>`.
  - If `<conj>` consists of only one `<atom>`, treat it as an `<atom>`.
  - Otherwise, connect the outputs of each `<atom>` to the inputs of an **AND** gate, one by one, from top to bottom.
- The rule of `<atom>` depends on different situations.
  - `<var>`: A variable, write down the name of the variable.
  - `<var> "'"`: A negation of a variable, write down the name of the variable, followed by a single quote.
  - `"(" <disj> ")"`: A sub-expression, draw the logic circuit diagram of the expression recursively.
  - `"(" <disj> ")'"`: A nagation of a sub-expression, draw the logic circuit diagram of the expression recursively, and connect its output to a **NOT** gate.

For example, the process of converting input `F=A(B+CD)'E+A'C+G` to a logic circuit diagram is shown below: first we obtain the main circuit `A(B+CD)'E+A'C+G`, and seperate it into three `<conj>`, `A(B+CD)'E`, `A'C`, and `G`.

- Seperate `A'(B+CD)'E` into three `<atom>`: `A'`, `(B+CD)'`, and `E`.
  - `A'` is a negation of a variable, write down its variable name, followed by a single quote.
  - `(B+CD)'` is a negation of a sub-expression, draw the logic circuit diagram of `B+CD` recursively, and connect its output to a **NOT** gate.
  - `E` is a variable, write down its variable name.
- Seperate `A'C` into two `<atom>`: `A'` and `C`.

    - `A'` is a negation of a variable, write down its variable name, followed by a single quote.
    - `C` is a variable, write down its variable name.
- `G` consists of only one `<atom>`, treat is as an `<atom>`. The `<atom>` is a variable, write down its variable name.

The converted logic circuit diagram is shown below:



After learning how to draw a logic circuit diagram, finally it's time to output the diagram to the screen with ASCII art! Below we will show you how to draw the logic gates first, then explain how we arrange the logic gates of the whole circuit.

First, the **OR** gates. For an **OR** gate with n inputs, it occupies a rectangular area that is $2n + 2$ rows high and 7 characters wide. In the 1-st row of the logic gates, the 2-nd, the 3-rd, and the 4-th characters are underscores (`_`). In the 2-nd row, the 1-st and 5-th characters are backslashes (`\`). In the 3-rd row, the 1-st character is a hyphen (`-`), and the 2-nd and the 6-th characters are backslashes (`\`). From the 4-th to the $(2n)$-th row, the 2-nd and 7-th characters are vertical bars (`|`); additionally, for all the odd-numbered rows within this range, the 1-st character is a hyphen (`-`). In the $(2n + 1)$-th row, the 1-st character is a hyphen (`-`), and the 2-nd and the 6-th characters are slashes (`/`). In the $(2n + 2)$-th row, the 1-st and the 5-th characters are slashes (`/`), and the 2-nd, the 3-rd, and the 4-th characters are underscores (`_`).

For example, below are the **OR** gates with 2, 3, and 4 inputs, respectively. The inputs of the logic gate are located on the left side of every odd-numbered row starting from the 3-rd row, while the output is located on the right side of the $(n + 2)$-th row. Please notice that the hyphens on the left are also a part of the logic gates.

```
 ___         ___         ___
\   \       \   \       \   \
-\   \      -\   \      -\   \
 |    |      |    |      |    |
-/   /      -|    |     -|    |
/___/        |    |      |    |
            -/   /      -|    |
            /___/        |    |
                        -/   /
                        /___/
```

Next, the **AND** gates. For an **AND** gates with n inputs, it occupies a rectangular area that is $2n + 2$ rows high and 7 characters wide. In the 1-st row of the logic gates, the 2-nd, the 3-rd, and the 4-th characters are underscores (`_`). In the 2-nd row, the 1-st character is a vertical bar (`|`), and the 5-th character is a backslash (`\`). In the 3-rd row, the 1-st character is a vertical bar (`|`), and the 6-th character is a backslash

(`\`). From the 4-th to the $(2n)$-th row, the 1-nd and 7-th characters are vertical bars (`|`). In the $(2n+1)$-th row, the 1-st character is a vertical bar (`|`), and the 6-th character is a slash (`/`). In the $(2n+2)$-th row, the 1-st character is a vertical bar (`|`), the 2-nd, the 3-rd, and the 4-th characters are underscores (`_`), and the 5-th character is a slash (`/`).

For example, below are the **AND** gates with 2, 3, and 4 inputs, respectively. The inputs of the logic gate are located on the left side of every odd-numbered row starting from the 3-rd row, while the output is located on the right side of the $(n+2)$-th row.

```
 ___         ___         ___
|   \       |   \       |   \
|    \      |    \      |    \
|     |     |     |     |     |
|    /      |     |     |     |
|___/       |     |     |     |
           |    /      |     |
           |___/       |     |
                       |    /
                       |___/
```

Next, the **NOT** gates. A **NOT** gate occupies a rectangular area that is $2n+2$ rows high and 7 characters wide. In the 1-st row of the logic gates, the 2-nd and the 3-rd characters are underscores (`_`). In the 2-nd row, the 1-st character is a vertical bar (`|`), and the 4-th character is a backslash (`\`). In the third row, the 1-st character is a vertical bar (`|`), and the 5-th character is a backslash (`\`). In the 4-th row, the 1-st character is a vertical bar (`|`), the 6-th character is a right parenthesis (`)`), and the 7-th character is a lowercase letter `o`. In the 5-th row, the 1-st character is a vertical bar (`|`), and the 5-th character is a slash (`/`). In the 6-th row, the 1-st character is a vertical bar (`|`), the 2-nd and the 3-rd characters are underscores (`_`), and the 4-th character is a slash (`/`).

Below is a **NOT** gate. The input of the logic gate is located on the left side of the 4-rd row, while the output is located on the right side of the 4-th row.

```
 __
|  \
|   \
|    )o
|   /
|__/
```

Next we will introduce how we output a variable. Each variable or its negation occupies a rectangular area that is 1 row high and 4 characters wide. The 1-st character is the name of the variable. If it's a negation of a variable, the 2-nd character is a single quote (`'`); otherwise, a space. The 3-rd and the 4-th characters are hyphens (`-`).

For example, below are the variable `A` and its negation, the output is located on the right side.

```
A --    A'--
```

Finally, let's assemble the logic gates into a complete circuit. We draw the circuit recursively: first the logic gate itself, then draw the sub-circuit connecting to the logic gate one by one, from top to bottom. To determine whether there are overlapping areas during circuit drawing, we define the rectangular area occupied by a logic circuit as the smallest rectangle area that can cover all logic gates, variables, and variable negations.

For a logic gate with n inputs, the i-th input from top to bottom is handled as follows: if the input is a variable or a negated variable, the rectangle area is directly drawn to the left of the logic gate at the same height as the input. If the input is another circuit, the circuit should be drawn to the left of the logic gate, leaving $2n+3$ spaces between them for connections later. Next, determine the vertical position of the circuit. First, shift the circuit up or down so that the height of its output position matches the height of the i-th input of the logic gate. Then, adjust the height further if it overlaps with any other previously drawn circuit's rectangle area, move it down until there is no overlap. Finally we connect the circuit and the logic gate. If the circuit's output is at the same height as the logic gate's input, fill the $2n+3$ spaces between them with hyphens (`-`). If the heights of the circuit's output and the logic gate's input are different, fill the right side of the circuit's output with $2i$ hyphens (`-`) followed by 1 plus sign (`+`), and fill the left side of the logic gate's input with $2(n-i)+2$ hyphens (`-`) followed by 1 plus sign (`+`). The two plus signs should align in the same column, and then fill the spaces between the two plus signs with vertical bars (`|`).

After drawing the complete logic circuit, at the output of the circuit add 2 hyphens (`-`) and 1 space, followed by the variable `F` indicating the output of the entire circuit.

For example, when drawing the logic circuit diagram of `F=A'B+(CD)'`, we first draw an `OR` gate, then draw the circuit of `A'B` and `(CD)'`.

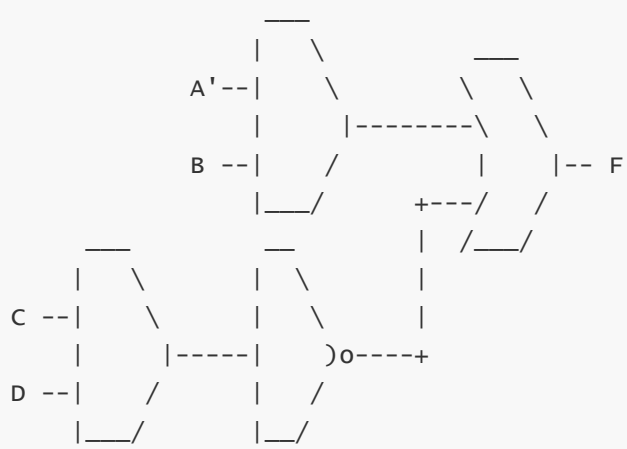
When drawing `A'B`, we first draw an `AND` gate. Since `A'` and `B` are the negation of a variable and another variable, we can connect them to the `AND` gate directly, obtaining the logic ciruit of `A'B`. The circuit diagram is shown below:

```
     ___
    |   \
A'--|    \
    |     |
B --|    /
    |___/
```

As for `(CD)'`, we first draw a `NOT` gate, then draw the `CD` part. Since `NOT` gate has only 1 input, there should be 5 spaces between the gate and the `CD` part. Since the output of the circuit has the same height as the input of the logic gate, fill the spaces with hyphens to connect the circuit. The circuit diagram of `(CD)'` is shown below:

```
      ___         __
     |   \       |  \
C  --|    \      |   \
     |     |-----|    )o
D  --|    /      |   /
     |___/       |__/
```

Finally, connect them one by one to the OR gate. There should be 7 spaces between `A'B` an the OR gate. Since the circuit's output is at the same height as the logic gate's input, fill the spaces with hyphens to connect them. As for `(CD)'`, there should be 7 spaces between `(CD)'` and the OR gate, and move down by 3 rows to prevent from overlapping with the previous drawn circuit. Connect the circuit with the OR gate according to the rules mentioned above. Finally, add the circuit output `F` and we're all done.



Now it's your turn to write a program to do the conversion. Good luck!

Input Format

The input only contains a single string S, the logical expression to be converted.

Output Format

The first line of the output contains two integer H and W, seperated by a space, indicating the number of rows and the number of characters on each row of the circuit diagram.

Followed by H lines, each containing W characters, indicating the converted logic circuit diagram.

Please note that your output must be the smallest rectangular area occupied by the logic circuit diagram, and all spaces must be output, including trailing spaces. Your program will be evaluated using strict comparison in this problem.

Constraints

- $3 \leq |S| \leq 100$
- Any input to the problem is given by an `<input>` non-terminal of the BNF grammar mentioned in the statement.

Subtasks

- Subtask 1 satisfies that $|S| \leq 5$.
- Subtask 2 satisfies that there will be no single quote (`'`) in S.
- Subtask 3 has no additional constraints.

Test Cases

Input 1

```
F=A'B+(CD)'
```

Output 1

```
12 41
                ___
               |   \          ___
           A'--|    \        \   \
               |     |--------\   \
           B --|    /          |    |-- F
               |___/      +---/   /
     ___        __        |  /___/
    |   \       |  \       |
C --|    \      |   \      |
    |     |-----|    )o----+
D --|    /      |   /
    |___/       |__/
```

Input 2

```
F=A(B+(C+D))'E+A'C+G
```

Output 2

```
14 73
                                          ___
                  ___         __             |   \
     ___         \   \       |  \        A --|    \           ___
    \   \     B ---\   \      |   \           |     |         \   \
C ---\   \         |    |-----|    )o---------|     |----------\   \
     |    |--------/   /      |   /           |     |           |    |
D ---/   /        /___/       |__/        E --|    /     +-----|    |-- F
    /___/                                     |___/      |     |    |
                                               ___       |G ---/   /
                                              |   \      |    /___/
                                          A'--|    \     |
                                              |     |----+
                                          C --|    /
                                              |___/
```

Input 3

```
F=((A+B)'(CD)+E)G((HI+J)+K')LM(OP)
```

Output 3

```
26 77
     ___          __
    \   \        |  \             ___
A ---\   \       |   \           |   \           ___
     |    |------|    )o--------|    \         \   \                    ___
B ---/   /       |   /           |     |--------\   \                  |   \
    /___/        |__/        +---|    /           |    |----------------|    \
                  ___        |  |___/     E ---/   /                   |     |
                 |   \       |                  /___/              G --|     |
            C --|    \      |                                         |     |
                 |     |----+                                +--------|     |
            D --|    /                                      |        |     |-- F
                 |___/                                      |    L --|     |
                  ___                                       |        |     |
                 |   \           ___                       |    M --|     |
            H --|    \         \   \           ___         |        |     |
                 |     |--------\   \         \   \        |     +--|    /
            I --|    /           |    |--------\   \       |     |  |___/
                 |___/      J ---/   /          |    |------+     |
                           /___/     K'---/   /                   |
                                          /___/                  |
                                           ___                   |
                                          |   \                  |
                                     O --|    \                  |
                                          |     |--------------+
                                     P --|    /
                                          |___/
```

Input 4

```
F=B+(B)+(B')'+((B)')'
```

Output 4

```
15 45
                                ___
                               \   \
                          B ---\   \
                 __             |    |
                |  \            B ---|    |
                |   \                |    |-- F
            B'--|    )o--------------|    |
                |   /                |    |
                |__/              +---/   /
     __          __               |  /___/
    |  \        |  \              |
    |   \       |   \             |
```

```
B --|    )o-----|    )o--------+
    |   /       |   /
    |__/        |__/
```

Input 5

```
F=X
```

Output 5

```
1 8
X ---- F
```

Illustrations

For Output 3, note that `B'` and `(B)'` are drawn differently according to the definition in BNF grammar.

Furthermore, please don't simplify the logic circuit or the logical expression. Convert the logical expression directly to the logic circuit diagram according to the process in the statement part.