The 18th Zhejiang University Programming Contest

Sponsored by

Contest Section

April 7, 2018

Problem List

A	Pretty Matrix
B	Liblume
C	Mergeable Stack
D	RAID-ZOJ
E	Crosses Puzzles
F	Schrödinger's Knapsack
G	Traffic Light
H	Boolean Expression
I	Honorifics
J	PPAP

This problem set should contain 10 (ten) problems on 18 (eighteen) numbered pages. Please inform a runner immediately if something is missing from your problem set.

Prepared by Zhejiang University ACM/ICPC Problem Setter Team.

Problem A. Pretty Matrix

Description

DreamGrid's birthday is coming. As his best friend, BaoBao is going to prepare a gift for him.

As we all know, BaoBao has a lot of matrices. This time he picks an integer matrix with n rows and m columns from his collection, but he thinks it's not pretty enough. On the one hand, he doesn't want to be stingy, but some integers in the matrix seem to be too small. On the other hand, he knows that DreamGrid is not good at algebra and hates large numbers, but some integers in the matrix seem to be too large and are not suitable for a gift to DreamGrid.

Based on the above consideration, BaoBao declares that a matrix is pretty, if the following conditions are satisfied:

1. For every integer a_{ij} in the matrix, $a_{ij} \geq A$.
2. For every integer a_{ij} in the matrix, $a_{ij} \leq B$.

where a_{ij} is the integer located at the i-th row and the j-th column in the matrix, and A and B are two integers chosen by BaoBao.

Given the matrix BaoBao picks, along with the two integers A and B, please help BaoBao change some integers in the matrix (BaoBao can change an integer in the matrix to any integer) so that the matrix becomes a pretty matrix. As changing integers in the matrix is tiring, please tell BaoBao the minimum number of integers in the matrix he has to change to make the matrix pretty.

Input

There are multiple test cases. The first line of input is an integer T (about 100), indicating the number of test cases. For each test case:

The first line contains four integers n, m, A and B. ($1 \leq n, m \leq 100, 1 \leq A, B \leq 10^5$). Their meanings are described above.

For the next n lines, the i-th line contains m integers $a_{i1}, a_{i2}, \ldots, a_{im}$ ($1 \leq a_{ij} \leq 10^5$), representing the original matrix.

Output

For each test case output one line indicating the answer. If it's impossible to make the matrix pretty, print "No Solution" (without quotes) instead.

Example

standard input	standard output
2	2
3 4 2 3	No Solution
3 2 2 2	
2 1 2 3	
2 3 100 3	
2 1 2 1	
1	
2	

Problem B. Liblume

Description

DreamGrid has a matrix A consisting of lowercase English letters. You are allowed to rearrange its rows any number of times (including zero times). Please calculate the maximum possible area of a submatrix in which every row and column is a palindromic string after the rearrangements.

Let's assume that the rows of matrix A are numbered from 1 to n from top to bottom and the columns are numbered from 1 to m from left to right. A matrix cell on the intersection of the i-th row and the j-th column can be represented as (i, j).

A palindromic string is a string that can be read the same way from left to right and from right to left. For example, "abacaba", "z", "abba" are palindromes.

A submatrix of matrix A is a group of four integers d, u, l, r $(1 \le d \le u \le n, 1 \le l \le r \le m)$. The submatrix contains all the cells (i, j) satisfying both $d \le i \le u$ and $l \le j \le r$. The area of the submatrix is the number of cells it contains.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and m $(1 \le n \times m \le 2 \times 10^5)$ – the number of rows and columns of the matrix.

Each of the next n lines contains m characters denoting the matrix.

It is guaranteed that the sum of $n \times m$ in all cases does not exceed 2×10^6.

Output

For each test case, output an integer denoting the maximum possible area of a submatrix in which every row and column is a palindromic string after the rearrangements.

Example

standard input	standard output

```
4
2 2
aa
aa
3 3
aaa
aaa
bbb
3 3
abc
def
ghi
3 3
aba
bab
aba
```
	```
	4
	9
	1
	9
	```

Problem C. Mergeable Stack

Description

Given n initially empty stacks, there are three types of operations:

- 1 s v: Push the value v onto the top of the s-th stack.
- 2 s: Pop the topmost value out of the s-th stack, and print that value. If the s-th stack is empty, pop nothing and print "EMPTY" (without quotes) instead.
- 3 s t: Move every element in the t-th stack onto the top of the s-th stack in order.

  Precisely speaking, denote the original size of the s-th stack by $S(s)$, and the original size of the t-th stack by $S(t)$. Denote the original elements in the s-th stack from bottom to top by $E(s,1), E(s,2), \dots, E(s,S(s))$, and the original elements in the t-th stack from bottom to top by $E(t,1), E(t,2), \dots, E(t,S(t))$.

  After this operation, the t-th stack is emptied, and the elements in the s-th stack from bottom to top becomes $E(s,1), E(s,2), \dots, E(s,S(s)), E(t,1), E(t,2), \dots, E(t,S(t))$. Of course, if $S(t) = 0$, this operation actually does nothing.

There are q operations in total. Please finish these operations in the input order and print the answer for every operation of the second type.

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and q ($1 \le n, q \le 3 \times 10^5$), indicating the number of stacks and the number of operations.

The first integer of the following q lines will be op ($1 \le op \le 3$), indicating the type of operation.

- If $op = 1$, two integers s and v ($1 \le s \le n$, $1 \le v \le 10^9$) follow, indicating an operation of the first type.
- If $op = 2$, one integer s ($1 \le s \le n$) follows, indicating an operation of the second type.
- If $op = 3$, two integers s and t ($1 \le s, t \le n$, $s \ne t$) follow, indicating an operation of the third type.

It's guaranteed that neither the sum of n nor the sum of q over all test cases will exceed 10^6.

Output

For each operation of the second type output one line, indicating the answer.

Example

standard input	standard output

2
2 15
1 1 10
1 1 11
1 2 12
1 2 13
3 1 2
1 2 14
2 1
2 1
2 1
2 1
2 1
3 2 1
2 2
2 2
2 2
3 7
3 1 2
3 1 3
3 2 1
2 1
2 2
2 3
2 3
	13
	12
	11
	10
	EMPTY
	14
	EMPTY
	EMPTY
	EMPTY
	EMPTY
	EMPTY
	EMPTY

Problem D. RAID-ZOJ

Description

It's a beautiful day outside. Birds are singing. Flowers are blooming. You turn on your computer and are ready for a day's work. But, alas, your disk **FAILED** and all your data goes down the drain! But don't worry, it's time for RAID (Redundant Array of Independent Disks) to come to your rescue!

The idea behind RAID is quite simple: make copies of data on other disks. This is exactly what RAID1 does. RAID1 makes a mirror for every disk, so each bit of your data is stored identically on two different disks. If one disk fails, you can still restore your data from another disk.

RAID1 is nice, but isn't clever enough, as you can only use 50% of your disks to store meaningful data. RAID3 is a smarter approach. Let's say you have m data disks, each containing n bits, where the i-th bit in the j-th data disk (note that we count i from 0 to $(n-1)$, and j from 0 to $(m-1)$) is indicated as $b_{i,j}$. RAID3 only requires you to buy one more disk, called the **row parity disk**, to store extra data. Let the i-th bit in the row parity disk to be r_i. RAID3 calculates r_i as follows:

$$r_i = b_{i,0} \oplus b_{i,1} \oplus \cdots \oplus b_{i,m-2} \oplus b_{i,m-1}$$

where $\oplus$ is the bitwise exclusive or operation ($0 \oplus 0 = 1 \oplus 1 = 0$, $0 \oplus 1 = 1 \oplus 0 = 1$). With the row parity disk in hand, if the j-th data disk fails, we can restore the data on that disk by adjusting the above formula to

$$b_{i,j} = r_i \oplus b_{i,0} \oplus b_{i,1} \oplus \cdots \oplus b_{i,j-2} \oplus b_{i,j-1} \oplus b_{i,j+1} \oplus b_{i,j+2} \oplus \cdots \oplus b_{i,m-2} \oplus b_{i,m-1}$$

What's more, you can now use $\frac{m}{m+1}$ of your disks to store meaningful data! Horray!

RAID3 seems to be such a perfect approach, that we even put ourselves under the illusion that we can sit back and relax after deploying RAID5 (similar to RAID3, but more reliable) on ZOJ and stop thinking about disk failures. But guess what, the careless administrator of the data center didn't notice the warning when a disk failure occurred, and when another disk failed in May 2016, everything was too late.

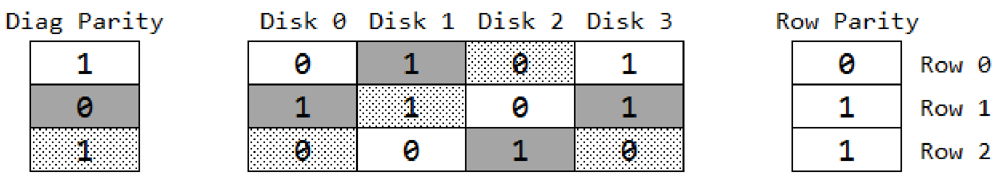
RAID3 and RAID5 can both protect their users from **one** disk failure, but if **two** disks fail simultaneously (though the probability is small) or you have a careless data center administrator, they still can't help you restore your data. This is why we invent RAID-ZOJ.

RAID-ZOJ is another RAID approach hoping to guard its users against two simultaneous disk failures. To achieve this goal, apart from the row parity disk, we add a second parity disk to the system, called the **diagonal parity disk**.

For a bit $b_{i,j}$ on the data disk, we define its diagonal value $D(b_{i,j}) = (i+j) \mod n$. Let d_i be the i-th bit on the diagonal parity disk, we calculate d_i as follows:

$$d_i = \bigoplus_{D(b_{k,j})=i} b_{k,j}$$

which means that d_i is calculated as the bitwise exclusive or of every bit whose diagonal value equals i. To help you understand, check the sample figure showing a RAID-ZOJ system with $n = 3$ and $m = 4$ below.



Now comes the final problem: is RAID-ZOJ capable of restoring data on two failed disks? We now provide you with a RAID-ZOJ system with m data disks, each containing n bits, a row parity disk, and a diagonal parity disk. In this system, exactly two data disks are broken. Your task is to restore the data on the two failed data disks so that the restored data are consistent with the two parity disks. As there might be multiple valid answers, you just need to calculate the minimum and the maximum possible number of 1s in the broken bits.

Input

There are multiple test cases. The first line of input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n \times m \le 10^6$), indicating the number of bits in each data disk and the number of data disks.

Each of the next n lines contains m characters denoting the bits in the data disks. The j-th character on the i-th line (don't forget we count both i and j from 0) is $b_{i,j}$ ($b_{i,j} \in \{0, 1, \text{X}\}$), indicating the i-th bit in the j-th data disk, where $b_{i,j} = \text{X}$ (ASCII code: 88) means that this bit is broken and its your task to restore its value.

The next line contains n characters $r_0, r_1, \dots, r_{n-1}$ ($r_i \in \{0, 1\}$), indicating the bits in the row parity disk.

The next line contains n characters $d_0, d_1, \dots, d_{n-1}$ ($d_i \in \{0, 1\}$), indicating the bits in the diagonal parity disk.

It's guaranteed that there exist exactly two different integers a and b such that $1 \le a, b \le m$ and $b_{i,a} = b_{i,b} = \text{X}$ for all $1 \le i \le n$.

It's also guaranteed that the sum of $n \times m$ over all test cases will not exceed 3×10^6.

Output

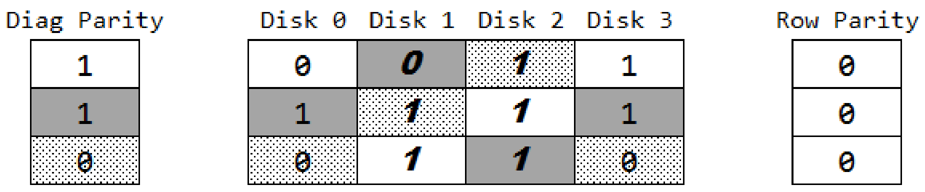
For each test case output one line containing two integers separated by a space, indicating the minimum and the maximum possible number of 1s in the broken bits. If no valid answer exists, print "No Solution" (without quotes) instead.

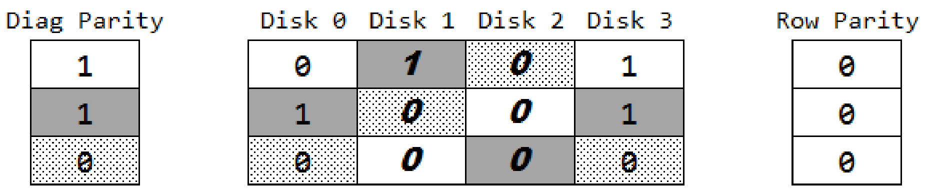
Example

standard input	standard output
4	1 5
3 4	0 4
0XX1	No Solution
1XX1	1 1
0XX0	
000	
110	
2 3	
X1X	
X0X	
10	
01	
3 5	
X0X01	
X0X01	
X0X01	
101	
010	
1 2	
XX	
1	
1	

Note

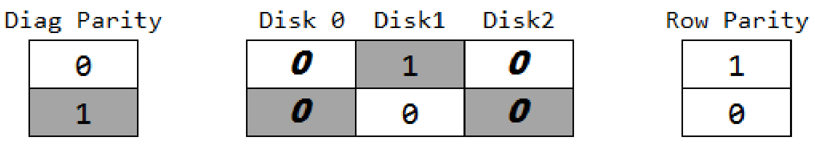
The two possible answers for the first sample test case are shown as follows (the restored broken bits are shown in bold):

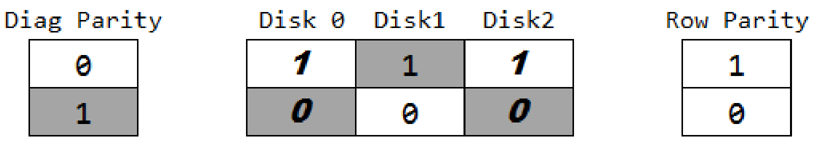


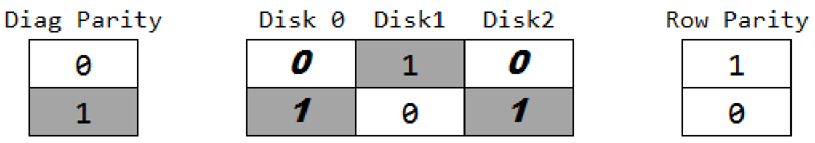


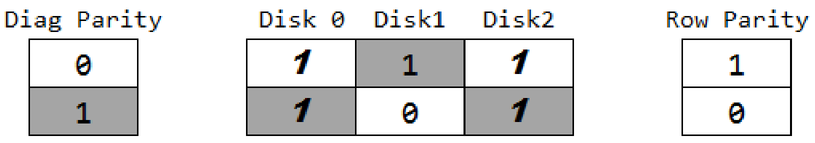
The first possible answer has five 1s in the broken bits, while the second possible answer has one 1. So the answer is “1 5”.

The four possible answers for the second sample test case are shown as follows (the restored broken bits are shown in bold):









The first possible answer has no 1 in the broken bits, while the second and the third possible answer each has two 1s, and the fourth possible answer has four 1s. So the answer is “0 4”.

Problem E. Crosses Puzzles

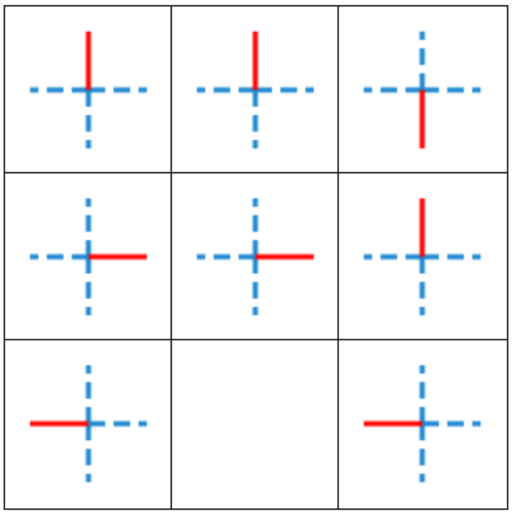
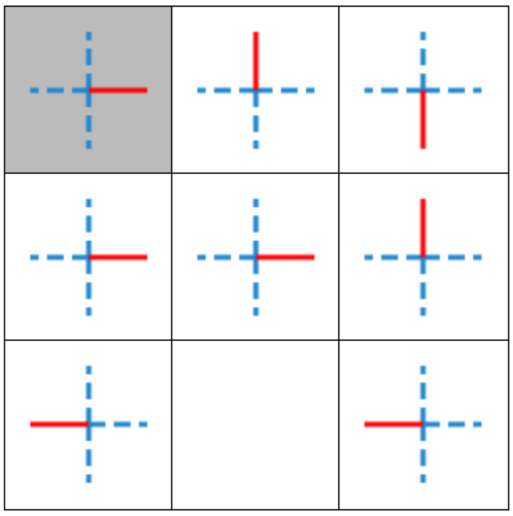
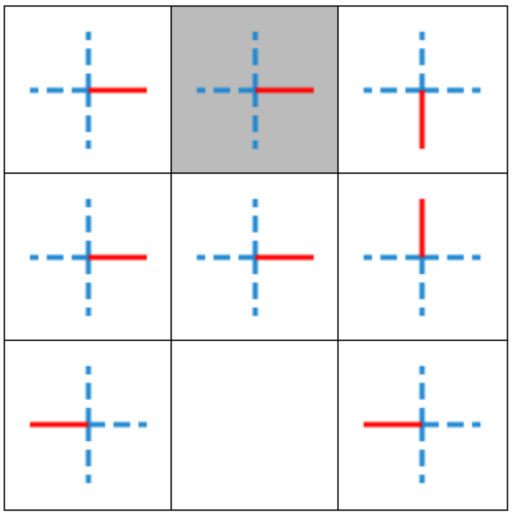
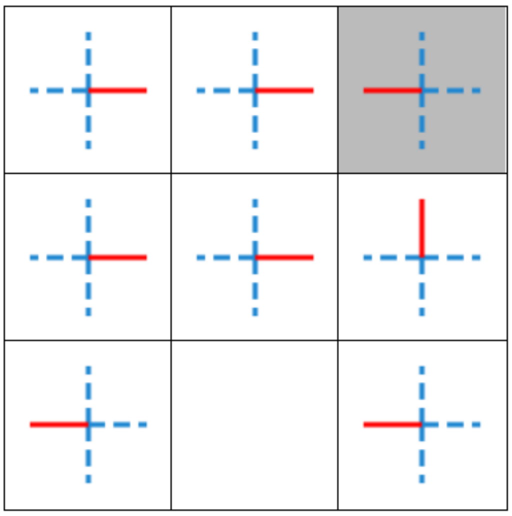
Description

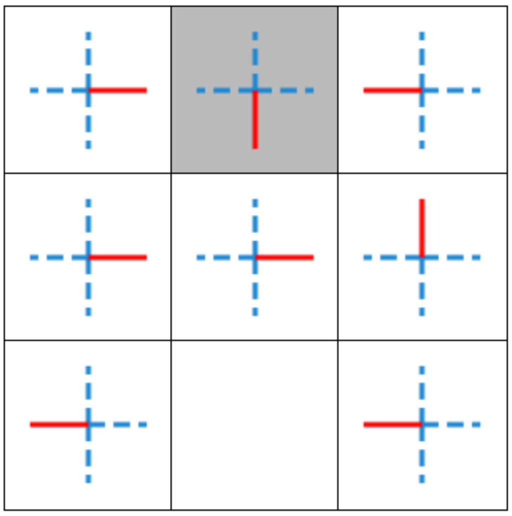
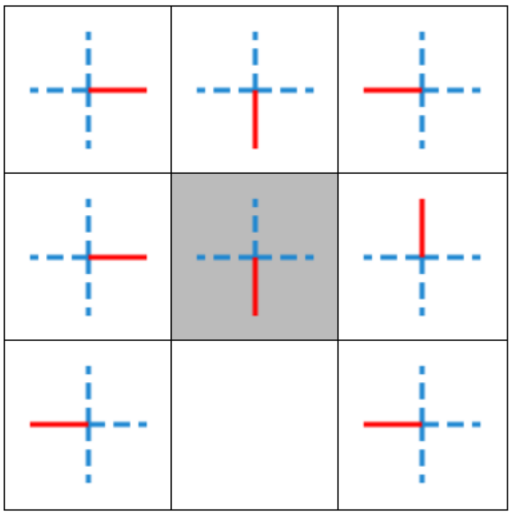
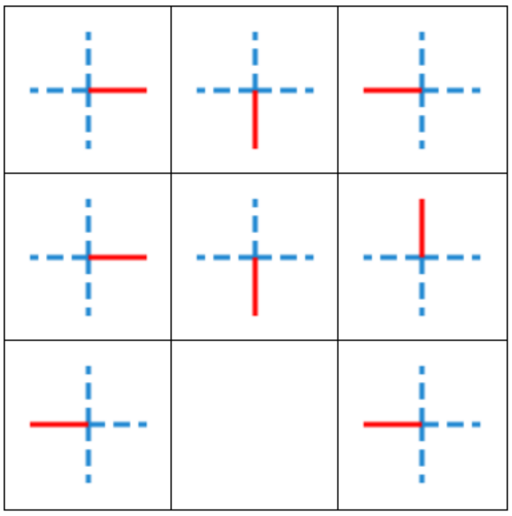
Given a grid with n rows and m columns, each cell may contain nothing or a cross consisting of three dotted segments and one solid segment. Let's denote the cell located on the i-th row and the j-th column by (i, j).

When you click on a cross, it will rotate 90 degrees clockwise, thus changing the direction the solid segment points to. Your goal is to click on some crosses in some proper order so that the solid segment of every cross points upwards.

But beware, this task is harder than you think it may be. After the rotation of a cross in (i, j) ends, the cross (i', j') pointed by the solid segment of (i, j) will also begin to rotate 90 degrees clockwise, and after the rotation of (i', j') ends, another cross may be triggered if pointed by the solid segment of (i', j')... This chain reaction comes to an end only when the solid segment of the previously-rotated cross points to an empty cell or points out of the map.

To help you understand, please check the following pictures showing a chain reaction on a grid with $n = m = 3$ step by step.

This is the initial state before the chain reaction. Now let's click on the cross $(1, 1)$. 	After clicking, $(1, 1)$ rotates 90 degrees clockwise and its solid segment points to $(1, 2)$ 
$(1, 2)$ is activated and rotates 90 degrees clockwise. After rotation, its solid segment points to $(1, 3)$. 	$(1, 3)$ is activated and rotates 90 degrees clockwise. After rotation, its solid segment points back to $(1, 2)$. 

$(1, 2)$ is activated again and rotates. After rotation, its solid segment points to $(2, 2)$. 	$(2, 2)$ is activated and rotates 90 degrees clockwise. After rotation, its solid segment points to $(2, 3)$. 
As $(2, 3)$ is an empty cell, the chain reaction stops. 	

Given a grid with n rows and m columns, please find out a proper order to click on the crosses so that the solid segment of every cross points upwards. You can perform at most 6000 clicks for each test case.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 50$), indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n, m \le 10$), indicating the size of the grid.

For the following n lines, the i-th line contains m integers $t_{i,1}, t_{i,2}, \dots, t_{i,m}$ ($-1 \le t_{i,j} \le 3$), where $t_{i,j}$ indicates the initial state of the cross at intersection (i, j).

- If $t_{i,j} = 0$, the solid segment of the cross in cell (i, j) is initially pointing upwards.
- If $t_{i,j} = 1$, the solid segment of the cross in cell (i, j) is initially pointing leftwards.
- If $t_{i,j} = 2$, the solid segment of the cross in cell (i, j) is initially pointing downwards.
- If $t_{i,j} = 3$, the solid segment of the cross in cell (i, j) is initially pointing rightwards.
- If $t_{i,j} = -1$, cell (i, j) is an empty cell.

Output

For each test case, first output one line containing one integer k ($0 \le k \le 6000$), indicating the number of clicks. Then k lines follow. Each line contains two integers i, j ($1 \le i \le n, 1 \le j \le m$) separated by a space, indicating that you click on the cross in cell (i, j).

If there's no way to make the solid segment of every cross point upwards in 6000 clicks, just output "-1" (without quotes) instead.

Example

standard input	standard output
2	8
3 2	1 1
2 3	1 2
-1 0	1 2
3 2	2 2
2 2	2 2
3 2	3 1
0 1	3 1
	3 2
	5
	1 1
	2 1
	2 1
	2 1
	1 2

Note

We've also prepared a playable example on the webpage. You can check that example for further understanding.

Problem F. Schrödinger's Knapsack

Description

DreamGrid has a magical knapsack with a size capacity of c called the Schrödinger's knapsack (or S-knapsack for short) and two types of magical items called the Schrödinger's items (or S-items for short). There are n S-items of the first type in total, and they all have a value factor of k_1; While there are m S-items of the second type in total, and they all have a value factor of k_2. The size of an S-item is given and is certain. For the i-th S-item of the first type, we denote its size by $s_{1,i}$; For the i-th S-item of the second type, we denote its size by $s_{2,i}$.

But the value of an S-item remains uncertain until it is put into the S-knapsack (just like Schrödinger's cat whose state is uncertain until one opens the box). Its value is calculated by two factors: its value factor k, and the remaining size capacity r of the S-knapsack just **after** it is put into the S-knapsack. Knowing these two factors, the value v of an S-item can be calculated by the formula $v = kr$.

For a normal knapsack problem, the order to put items into the knapsack does not matter, but this is not true for our Schrödinger's knapsack problem. Consider an S-knapsack with a size capacity of 5, an S-item with a value factor of 1 and a size of 2, and another S-item with a value factor of 2 and a size of 1. If we put the first S-item into the S-knapsack first and then put the second S-item, the total value of the S-items in the S-knapsack is $1 \times (5-2) + 2 \times (3-1) = 7$; But if we put the second S-item into the S-knapsack first, the total value will be changed to $2 \times (5-1) + 1 \times (4-2) = 10$. The order does matter in this case!

Given the size of DreamGrid's S-knapsack, the value factor of two types of S-items and the size of each S-item, please help DreamGrid determine a proper subset of S-items and a proper order to put these S-items into the S-knapsack, so that the total value of the S-items in the S-knapsack is maximized.

Input

The first line of the input contains an integer T (about 500), indicating the number of test cases. For each test case:

The first line contains three integers k_1, k_2 and c ($1 \le k_1, k_2, c \le 10^7$), indicating the value factor of the first type of S-items, the value factor of the second type of S-items, and the size capacity of the S-knapsack.

The second line contains two integers n and m ($1 \le n, m \le 2000$), indicating the number of the first type of S-items, and the number of the second type of S-items.

The next line contains n integers $s_{1,1}, s_{1,2}, \dots, s_{1,n}$ ($1 \le s_{1,i} \le 10^7$), indicating the size of the S-items of the first type.

The next line contains m integers $s_{2,1}, s_{2,2}, \dots, s_{2,m}$ ($1 \le s_{2,i} \le 10^7$), indicating the size of the S-items of the second type.

It's guaranteed that there are at most 10 test cases with their $\max(n, m)$ larger than 100.

Output

For each test case output one line containing one integer, indicating the maximum possible total value of the S-items in the S-knapsack.

Example

standard input	standard output

3
3 2 7
2 3
4 3
1 3 2
1 2 10
3 4
2 1 2
3 2 3 1
1 2 5
1 1
2
1
	23
	45
	10

Note

For the first sample test case, you can first choose the 1st S-item of the second type, then choose the 3rd S-item of the second type, and finally choose the 2nd S-item of the first type. The total value is $2 \times (7-1) + 2 \times (6-2) + 3 \times (4-3) = 23$.

For the second sample test case, you can first choose the 4th S-item of the second type, then choose the 2nd S-item of the first type, then choose the 2nd S-item of the second type, then choose the 1st S-item of the second type, and finally choose the 1st S-item of the first type. The total value is $2 \times (10-1) + 1 \times (9-1) + 2 \times (8-2) + 2 \times (6-3) + 1 \times (3-2) = 45$.

The third sample test case is explained in the description.

It's easy to prove that no larger total value can be achieved for the sample test cases.

Problem G. Traffic Light

Description

DreamGrid City is a city with $n \times m$ intersections arranged into a grid of n rows and m columns. The intersection on the i-th row and the j-th column can be described as (i, j), and two intersections (i_1, j_1) and (i_2, j_2) are connected by a road if $|i_1 - i_2| + |j_1 - j_2| = 1$.

At each intersection stands a traffic light. A traffic light can only be in one of the two states: 0 and 1. If the traffic light at the intersection (i, j) is in state 0, one can only move from (i, j) to $(i + 1, j)$ or $(i - 1, j)$; If the traffic light is in state 1, one can only move from (i, j) to $(i, j + 1)$ or $(i, j - 1)$ (of course, the destination must be another intersection in the city).

BaoBao lives at the intersection (s_i, s_j), and he wants to visit his best friend DreamGrid living at the intersection (f_i, f_j). After his departure, in each minute the following things will happen in order:

- BaoBao moves from his current intersection to another neighboring intersection along a road. As a law-abiding citizen, BaoBao has to obey the traffic light rules when moving.
- Every traffic light changes its state. If a traffic light is in state 0, it will switch to state 1; If a traffic light is in state 1, it will switch to state 0.

As an energetic young man, BaoBao doesn't want to wait for the traffic lights, and he **must** move in each minute until he arrives at DreamGrid's house. Please tell BaoBao the shortest possible time he can move from (s_i, s_j) to (f_i, f_j) to meet his friend, or tell him that this is impossible.

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first line contains two integers n and m $(1 \le n \times m \le 10^5)$, indicating the size of the city.

For the following n lines, the i-th line contains m integers $t_{i,1}, t_{i,2}, \dots, t_{i,m}$ $(0 \le t_{i,j} \le 1)$, where $t_{i,j}$ indicates the initial state of the traffic light at intersection (i, j).

The next line contains four integers s_i, s_j, f_i and f_j $(1 \le s_i, f_i \le n,\ 1 \le s_j, f_j \le m)$, indicating the starting intersection and the destination intersection.

It's guaranteed that the sum of $n \times m$ over all test cases will not exceed 3×10^5.

Output

For each test case output one line containing one integer, indicating the shortest possible time (in minute) BaoBao can move from (s_i, s_j) to (f_i, f_j) without stopping. If it is impossible for BaoBao to arrive at DreamGrid's house, print "-1" (without quotes) instead.

Example

standard input	standard output
4	3
2 3	5
1 1 0	-1
0 1 0	0
1 3 2 1	
2 3	
1 0 0	
1 1 0	
1 3 1 2	
2 2	
1 0	
1 0	
1 1 2 2	
1 2	
0 1	
1 1 1 1	

Note

For the first sample test case, BaoBao can follow this path: $(1, 3) \to (2, 3) \to (2, 2) \to (2, 1)$.

For the second sample test case, due to the traffic light rules, BaoBao can't go from $(1, 3)$ to $(1, 2)$ directly. Instead, he should follow this path: $(1, 3) \to (2, 3) \to (2, 2) \to (2, 1) \to (1, 1) \to (1, 2)$.

For the third sample test case, it's easy to discover that BaoBao can only go back and forth between $(1, 1)$ and $(1, 2)$.

Problem H. Boolean Expression

Description

Given a valid boolean expression consisting of seven types of characters 'T' (true), 'F' (false), '!' (not), '&' (and), '|' (or), '(' (left bracket) and ')' (right bracket), you can add any number of these seven types of characters into the expression to make it true (of course, the expression must still be valid).

Please calculate the minimum number of characters you have to add into the expression to make it true.

The priority of the boolean operators are: '!' > '&' > '|', just the same as their priority in most of the programming languages.

To be specific, a valid boolean expression can be expressed by the following BNF,

`<bool> ::= 'T' | 'F' | '(' <expr> ')'`

`<not-expr> ::= '!' <not-expr> | <bool>`

`<and-expr> ::= <and-expr> '&' <not-expr> | <not-expr>`

`<expr> ::= <expr> '|' <and-expr> | <and-expr>`

where `<expr>` is a valid boolean expression.

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first and only line contains a boolean expression s ($1 \le |s| \le 5 \times 10^5$) without space consisting of 'T', 'F', '!', '&', '|', '(' and ')'.

It's guaranteed that the given expression is valid, and the sum of $|s|$ over all test cases will not exceed 5×10^6.

Output

For each test case output one line containing one integer, indicating the minimum number of characters you have to add into the expression to make it true.

Example

standard input	standard output
4 !T&F T&!F|T&F (T&(!T|(!!F))&T) !!!!!F	2 0 1 0

Note

For the first sample test case, we can add a pair of brackets and change the expression to !(T&F).

For the second sample test case, note that the priority of '&' is higher than '|', so the original expression is already true.

For the third sample test case, we can add a '!' and change the expression to !(T&(!T|(!!F))&T).

We kindly remind you that the stack size of the online judge system is limited to 8M, so please use your stack space wisely.

Problem I. Honorifics

Description

An honorific is a title that conveys esteem or respect for position or rank when used in addressing or referring to a person. Sometimes, the term "honorific" is used in a more specific sense to refer to an honorary academic title. It is also often conflated with systems of honorific speech in linguistics, which are grammatical or morphological ways of encoding the relative social status of speakers.

In Japenese, "-san" (sometimes pronounced as "han" in Kansai dialect) is the most commonplace honorific and is a title of respect typically used between equals of any age. Although the closest analogs in English are the honorifics "Mr.", "Miss", "Ms.", or "Mrs.", "-san" is almost universally added to a person's name; "-san" can be used in formal and informal contexts and for any gender. Because it is the most common honorific, it is also the most often used to convert common nouns into proper ones.

In Korean, "-ssi" is the most commonly used honorific used amongst people of approximately equal speech level. It is attached at the end of the full name, such as Gim Cheolsu-ssi or simply after the first name, Cheolsu-ssi, if the speaker is more familiar with someone. Appending "-ssi" to the surname, for instance Gim-ssi, can be quite rude, as it indicates that the speaker considers himself to be of a higher social status than the person he is speaking to.

In this problem, you are given a series of Japanese or Korean names, please add an honorific to each name. You should append "-san" to each Japanese name, and append "-ssi" to each Korean name.

In the input file, Japanese names are given in Kunrei-shiki romanization system, and Korean names are given in revised romanization system. Each name consists of two capitalized words composed of Latin alphabet. The first word is the surname (family name), and the second word is the forename (given name).

The test data contains 10000 randomly generated names. `train.txt` is the training data for you, which can be downloaded from the webpage of this problem, and its format is described in the input section. The training data only has 6000 names covering 60%∼ 70% surnames and forenames. Please pay attention to the generalization of your solution.

Since there may be some rare and very confused cases even for a human, your solution only needs to achieve at least 99% correct rate on the test.

Input

For the formal test, there are multiple test cases. The first line of input contains an integer T ($T = 10000$), indicating the number of test cases. For each test case:

The first line contains two words (the first character of each word is capitalized), which is either a Japanese name or a Korean name. The length of each word will not exceed 20.

For `train.txt`, the first line contains an integer T ($T = 6000$), indicating the number of training data. For each training data:

The first line contains two words (the first character of each word is capitalized), which is either a Japanese name or a Korean name followed by the correct honorific of this data. The length of each word (without the honorific) will not exceed 20.

Output

For each test case, output the name and the honorific. You should append "-san" to each Japanese name, and append "-ssi" to each Korean name. You need to achieve 99% correct rate on it at least.

Example

standard input	standard output
7 Fuzii Mina Nakamoto Yuuta Song Junggi Hirai Momo Seonu Jeonga Gong Yu Bang Mina	Fuzii Mina-san Nakamoto Yuuta-san Song Junggi-ssi Hirai Momo-san Seonu Jeonga-ssi Gong Yu-ssi Bang Mina-ssi

Note

Both the training data and the test data are generated as follows: for each case, toss a coin with an equally likely outcome to decide whether it is a Japanese name or a Korean name. If it is a Japanese name, then a surname and a forename are picked randomly from the Japanese surname list and Japanese forename list respectively; If it is a Korean name, then a surname and a forename are picked randomly from the Korean surname list and Korean forename list respectively.

In the final test data, the Japanese surname and forename list contain about 480 surnames and 2800 forenames respectively, and the Korean surname and forename list contain about 200 surnames and 2800 forenames respectively.

In the downloadable training data, the Japanese/Korean surnames/forenames list is a random subset of the corresponding one in the final test data.

These are no common surnames between the Japanese surname list and the Korean surname list, so a generated Japanese name and a generated Korean name will not be the same. However, there are a very small amount of common forenames between the Japanese forename list and the Korean forename list, so please be careful with these cases.

Reference:

https://en.wikipedia.org/wiki/Honorific

https://en.wikipedia.org/wiki/Japanese_honorifics#San

https://en.wikipedia.org/wiki/Korean_honorifics#-ssi

Problem J. PPAP

Description

"I have a pen. I have an apple. Uh! Applepen."

"I have a pen. I have pineapple. Uh! Pineapplepen."

The above lyrics are taken from *PPAP*, a single by Pikotaro. It was released as a music video on YouTube on 25 August 2016, and has since become a viral video. As of October 2017, the official video has been viewed over 126 million times.

Let's view this song from a mathematical perspective. In the lyrics there actually hides a function PPAP(a, b), which takes two lowercased string a and b as the input and works as follows:

- First, calculate $s_1 = b + a$ (+ here means string concatenation).
- Then, capitalize the first character of s_1 to get s_2.
- Make s_2 as its output, and the function is done.

For example, we have PPAP("pen", "apple") = "Applepen", and PPAP("pen", "pineapple") = "Pineapplepen".

Given two lowercased strings a and b, your task is to calculate PPAP(a, b).

Input

The first line of the input contains an integer T (about 100), indicating the number of test cases. For each test case:

The first and only line contains two strings a and b ($1 \le |a|, |b| \le 30$) separated by one space. It's guaranteed that both a and b consist of only lowercase English letters.

Output

For each test case output one line containing one string, indicating PPAP(a, b).

Example

standard input	standard output
3 pen apple pen pineapple abc def	Applepen Pineapplepen Defabc