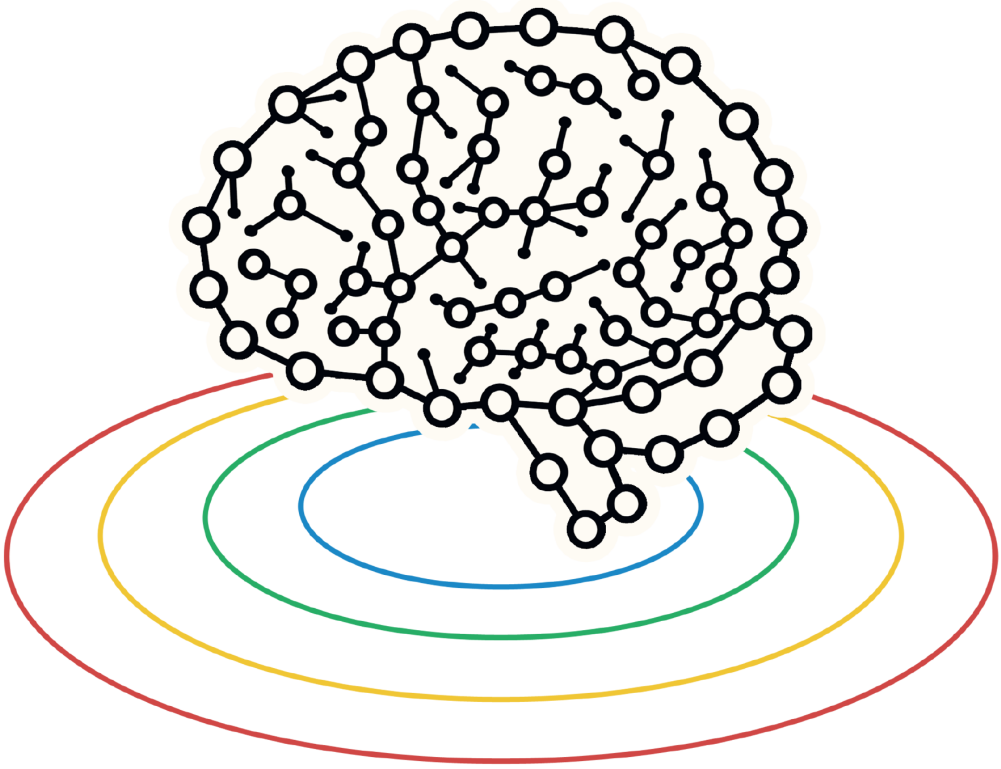
Association for Computing Machinery
University of the Philippines Diliman Student Chapter, Inc.



ALGOLYMPICS 2022

UP ACM PROGRAMMING COMPETITION

FINALS ROUND

Contents

Very important! Read the following:

- Your solution will be checked by running it against several hidden test cases. You will not have access to these cases, but a correct solution is expected to handle them correctly.
- The output checker is usually strict, so follow these guidelines strictly:
  - It is *space sensitive*. Do not output extra leading or trailing spaces. Do not output extra blank lines unless explicitly stated.
  - It is *case sensitive*. So, for example, if the problem asks for the output in lowercase, follow it.
  - Do not print any tabs. (No tabs will be required in the output.)
  - Do not output anything else aside from what's asked for in the Output section. So, do not print things like "Please enter t".
  - Generally speaking, each output line should be ended by a newline ('\n') character, even if there are no succeeding output lines. For example, in C/C++ this could look like `printf("The only line\n")`. Note that for Python, using `print("The only line")` by default adds the newline at the end.

  Not following the output format strictly and exactly may result in a *Wrong Answer* verdict.
- Do not read from, or write to, a file. You must read from the standard input and write to the standard output.
- For Java, do not add a package line at the top of your code. Otherwise, your submission will be judged *Runtime Error*.
- Only include one file when submitting: the source code (.cpp, .java, .py, etc.) and nothing else.
- Only use letters, digits and underscores in your filename. Do not use spaces, or other special symbols.
- Many problems have large input file sizes, so use fast I/O. For example:
  - In Java, use BufferedReader and PrintWriter.
  - In C/C++, use scanf and printf.

  The Elimination Round is a good time to learn these if you haven't yet.
- All problems are solvable in C++, and Java, but the same is not guaranteed for Python due to its slowness.
- Good luck and enjoy the contest!

Problem A

Abducted to the Sushiimon World, But At Least I Have My Smartphone

Time Limit: 1 second

One day, I was just minding my own business, crossing the street... when a truck came careening out of nowhere! The next thing I know, I'm floating in the void, clutching onto my smartphone for dear life... and then a bright light!

When I open my eyes, I am in an unknown land. One of the locals informs me that I am currently on the faraway planet of Sushii, inhabited by fantastical creatures called Sushiimons. With a lack of any other options, I am immediately conscripted into the Survey Corps of the expedition team and put in charge of capturing Sushiimons for the researchers.

Fortunately, the modern services on my smartphone allow me to easily capture many Sushiimons and rise through the ranks!

There are two kinds of Sushiimons that we can capture—Red and Blue. Each (Red and Blue) has an independent "Research Level", which starts at Level 0 and can go as high as Level n; your Research Level of some color increases by catching and submitting Sushiimons of that color. In particular, you are given an array $r_1, r_2, r_3, \ldots, r_n$, where r_i describes the number of Red Sushiimons that you need to submit in order to go from Red Research Level $i-1$ to Level i; another array $b_1, b_2, b_3, \ldots, b_n$ describes the similar requirements for advancing in Blue Research Levels. Your "Total Research Level" is equal to your Red Research Level plus your Blue Research Level.

As an additional restriction, the research team needs more and more Red Sushiimons to understand them sufficiently, so it is always the case that $r_1 \leq r_2 \leq r_3 \leq \cdots \leq r_n$ (but **no such restriction** is guaranteed for b).

For example, consider $r = \{1,1,2,2\}$ and $b = \{1,3,1,2\}$. I choose to catch 8 Sushiimons—4 Red and 4 Blue.

- I can reach Red Research Level 3 (Submit $1 \rightarrow$ Submit $1 \rightarrow$ Submit 2)
- I can reach Blue Research Level 2 (Submit $1 \rightarrow$ Submit 3).

Thus, my Total Research Level becomes 5. But actually, there is a way to reach Total Research Level 5 while catching *fewer* Sushiimons. Can you see it?

For each k from 1 to $2n$, answer the following question: Find the minimum number of Sushiimons that I need to catch in order to go from Total Research Level 0 to Total Research Level k. Each question should be considered and answered independently.

Input

The first line of input contains a single integer T, the number of test cases. T test cases follow. Each test case is described in the following manner.

First, a line containing a single integer n, the maximum Research Level for Red and Blue in this test case.

Next, a line containing n space-separated integers $r_1, r_2, r_3, \ldots, r_n$.

Finally, a line containing n space-separated integers $b_1, b_2, b_3, \ldots, b_n$.

Output

For each test case, output a line containing $2n$ space-separated integers, where the kth of these is the minimum number of Sushiimons that must be caught to reach Total Research level k.

Constraints

The sum of n across all test cases is $\leq 3 \times 10^5$
$1 \leq n$
$1 \leq r_i, b_i \leq 10^9$
$r_1 \leq r_2 \leq r_3 \leq \cdots \leq r_n$

Sample Input	Sample Output
1 4 1 1 2 2 1 3 1 2	1 2 3 5 7 9 11 13

Problem B
Alien Chess
Time Limit: 1 second

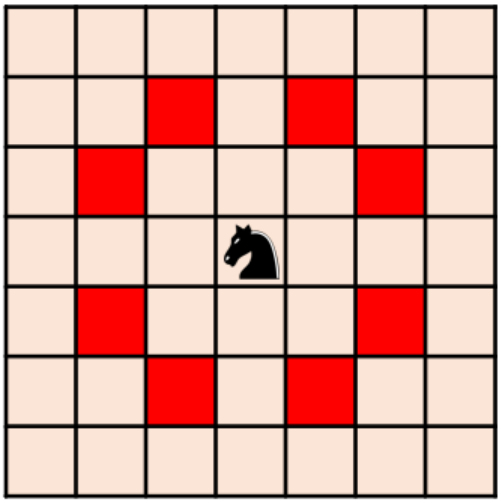
This is an interactive problem.

The aliens known as the Chessians have invaded Earth! Despite their superior technological firepower, they agree to leave humanity alone if we can prove our intelligence by besting them in a game of Alien Chess. You foolishly accept to be humanity's representative.

Unfortunately, the Chessian's representative is Alien Garry Kasparov. Oh, and also, Alien Chess is only superficially similar to Human Chess, and is, for the most part, a completely different game with a different set of rules. Oops?

Alien chess still takes place on an 8×8 chessboard. However, some of its squares have been destroyed!

Alien chess only has a single piece, a knight. The knight can move according to the same rules as in Human Chess—two steps in one direction, then one step in a perpendicular direction (see diagram). A knight may only move to squares that are not destroyed, and also it must remain on the chessboard at all times.



The rules are as follows. There are two roles—`First` and `Second`.

- To begin, `Second` places the knight on some non-destroyed square of their choice.
- The players take turns moving the knight to any of the non-destroyed squares that the knight can reach in one move. `First` makes the first move.
- Whenever the knight leaves a square, that square is immediately **destroyed**.
- If a player cannot make a valid move (when all valid squares the knight can move to from its current position have already been destroyed), that player **loses** the game.

Thankfully, the Chessians vastly underestimate you, so you are allowed two concessions.

1. After seeing the initial state of the chessboard, you can choose whether you want to be `First` or `Second`.
2. You are allowed to bring a computer with you to the game!

The fate of humanity rests on you being able to solve this problem! Good luck!

Interaction

Notation for the chessboard: Let (i, j) represent the square in the ith row from the top and jth column from the left, for $1 \leq i \leq 8$ and $1 \leq j \leq 8$. **Please take note of this coordinate system**.

In interactive problems, you communicate with an interactor. You send messages to the interactor by printing to standard output. You receive responses from the interactor by reading from standard input.

First, you will be given the initial state of the chessboard. Read 8 lines, each containing a string of 8 characters. The jth character in the ith line will be `.` if (i, j) is not destroyed, and `#` if it is destroyed. At least one cell in this chessboard will not be destroyed.

Now, output a line containing either `First` or `Second`, depending on which one **you** want to be.

- If you pick `First`, you should then read a line with two space-separated integers i_s and j_s, meaning that the enemy has placed the knight on (i_s, j_s).
- If you pick `Second`, you instead print a line with two space-separated integers i_s and j_s, meaning that you choose to place the knight on (i_s, j_s). This square should not be destroyed.

From this point on, `First` and `Second` will take turns making their move. `First` goes first. On your turns, output your move. On your opponent's turns, read their move.

Suppose that the knight is currently on (i, j).

- A move consists of a line containing two space-separated integers di and dj, where $(di, dj) \in \{(-2,-1),(-2,1),(-1,-2),(-1,2),(1,-2),(1,2),(2,-1),(2,1)\}$. This means the turn player wishes to move the knight from its current position of (i, j) to the square $(i+di, j+dj)$.
- If the turn player has no valid moves, they must output `0 0` instead.

This is valid only if $(i+di, j+dj)$ is within the chessboard, and also not destroyed. Also, remember that square (i, j) is immediately destroyed after making this move.

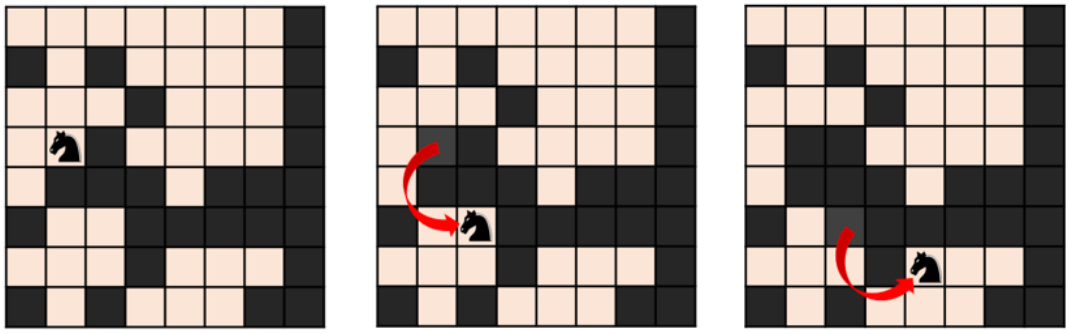
If any of your responses are invalid, or if you are forced to print `0 0`, the judge will immediately send `LOSE`, and then send no further messages. Your program should terminate. You lose.

If the judge is forced to print `0 0`, then your program will be accepted as correct. The judge will then send no further messages; your program should terminate.

Sample Interaction

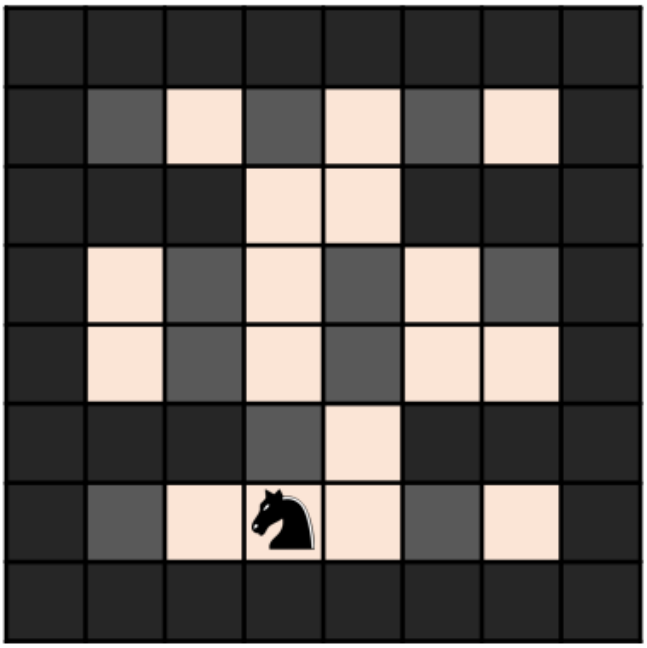
```
Judge                   You
.......#
#.#....#
...#...#
..#....#
.###.###
#..#####
...#...#
#.#...##
                        Second
                        4 2
2 1
                        1 2
0 0
```

This first sample interaction corresponds to the following sequence of moves.



```
Judge                   You
########
#......#
###..###
#......#
#......#
###..###
#......#
########
                        First
2 2
                        2 1
-2 1
                        2 1
-2 1
                        2 1
1 -2
                        2 1
-1 -2
                        1 -2
-2 1
                        2 1
0 0
```

This second sample interaction ends on the following board state.



Judge	You
########	
########	
#.######	
########	
########	
########	
########	
########	
	Second
	3 2
0 0	

The third sample interaction is over before `First` can make a move.

Problem C
Coin Sort
Time Limit: 2 seconds

The Bank of Sigma Polaris (or BSP for short) is in charge of the distribution of coinage within the Sigma Polaris solar system. Recently, they've minted new coins with brand new Space-Patriotic designs. However, through an unfortunate error, they have mixed up the designs of the 10 Zollar coin and the 5 Zollar coin, making them completely identical except for the number! This has led to the confusion of the system's inhabitants, and the coins, being already under wide circulation, cannot be recalled for at least several decades.

To compensate for their error, the BSP issued a pocket coin sorter to all of their citizens, free of charge! The coin sorter takes a stack of coins and rearranges it through a series of move operations. One move operation takes a continuous subsegment of one or more coins *of the same denomination*, removes it from the stack, and appends the removed sequence either to the top or bottom of the stack. The goal is to have all coins of the same denomination contiguous to each other. That is, for every two 5 Zollar coins in the row, there are no 10 Zollar coins in between them, and vice versa. At that point, the coins are considered sorted.

For example, let's say we have the following Zollar coins, from top to bottom:

```
10 5 5 10 10 5
```

This can be sorted using a single move by picking up the 2nd and 3rd coins (which are both 5 Zollar coins) and appending them to the bottom end of the stack. The result will be:

```
10 10 10 5 5 5
```

Given several stacks of 5 and 10 Zollar coins, can you determine how many moves it will take, at the minimum, to sort them?

Input

The input begins with a line containing an integer T, the number of test cases. T test cases follow.

Each test case consists of two lines. The first line contains a positive integer N, the number of coins in the stack. The second line contains N space-separated integers representing the coins. The integers are either 5 and 10 corresponding to their denominations.

Due to huge input file size, it is recommended to use fast methods of input for this problem.

Output

For each test case, output a single line containing the minimum number of moves it would take to sort the given stack.

Constraints

$1 \leq T \leq 20$
$1 \leq N \leq 10^6$

Sample Input	Sample Output
2 6 10 5 5 10 10 5 8 10 10 5 10 5 5 10 5	1 2

SAMSUNG Research

This problem is contributed by Samsung R&D Institute Philippines

Problem D
Dimensional Elevator
Time Limit: 2 seconda

The Science Pub is a busy pub located in one of the mining belts of Orion Beta, where multitudes of people go to experience the infamous Janx Spirit[1]. The pub is composed of multiple dimensions, layered together like floors in a massive tower. On the first floor is a dimensional elevator, transporting people from floor to floor, one dimension at a time. On its first floor are N customers, and customer i needs to be seated on floor p_i. They have not yet boarded the elevator, but they might need to use it to get to the floor they need to be at!

The dimensional elevator is the only way of going between floors in the Science Pub. The elevator takes 3 seconds to go up or down by one floor. People on any floor may choose to enter the elevator, which is instantaneous. However, it takes a single person 3 seconds to leave the elevator, and people can only leave the elevator one-by-one.

With these conditions, what is the minimum amount of time (in seconds) needed for all customers to reach their corresponding floor?

Input

Input begins with a single line containing an integer T, the number of test cases. T test cases follow. Each test case is described in the following manner.

The first line of each case contains a single integer N, the number of customers.

The next line contains N space-separated integers, denoting p_1 to p_N, the corresponding seating floor of each customer.

Output

For each test case, output a single line containing the minimum amount of time, in seconds, needed for all customers on the first floor to reach their corresponding floor.

Constraints

$1 \leq T \leq 100$
$1 \leq N, p_i \leq 10^4$

Sample Input	Sample Output
2 3 2 2 3 4 2 4 2 7	15 30

Explanation

In the first test case of the sample input, we have $N = 3$ customers. Customers 1 and 2 want to go to floor 2, while customer 3 wants to go to floor 3. Customers 1, 2, and 3 board the elevator on the first floor, which is instantaneous. The elevator moves up once to the 2nd floor, which takes 3s. It unloads customers 1 and 2, taking 6s. Finally, it goes up another level and then unloads customer 3, adding another 6s. Altogether, it takes 15s to unload all the people to their corresponding floors, which is the minimum.

[1] An excerpt from the Hitchhiker's Guide to the Galaxy is an ancient Orion mining song:
"Oh don't give me none more of that Old Janx Spirit
No, don't you give me none more of that Old Janx Spirit
For my head will fly, my tongue will lie, my eyes will fry and I may die
Won't you pour me one more of that sinful Old Janx Spirit"

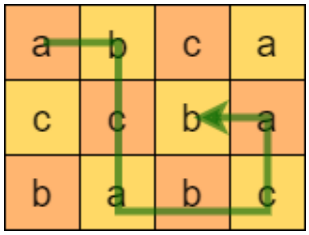
Problem E
Mysterious Symbols

Time Limit: 1 seconds

Deep in the ruins of planet Algodora, there lies several ancient floor mosaics of mystical origins. A mosaic is an $R \times C$ grid filled with mysterious symbols. According to legends, unlocking the secrets of these symbols will lead to incredible fortune!

Recently, you've discovered a hidden ancient text that could be a clue to solving the puzzle. According to the text, the symbols on the grid come from a set of K symbols which all follow a cyclic order. You need to traverse the grid by visiting each cell *at least once* using only horizontal and vertical single-step movements (i.e. moving to the cell directly up, down, left, or right).

Additionally, you must follow the cyclic order when stepping on the symbols in the grid. For example, given $K = 3$ symbols $a, b,$ and c which follow the cyclic order $a \to b \to c$, if you are currently on a cell with symbol b, you can only step on a cell containing symbol c. Likewise, if you are on symbol c, you can only step on a cell with symbol a. One example of a mosaic, along with a valid path, is shown below.



Note however, that you can start at any cell of the grid, and not necessarily on the first symbol of the cycle (e.g. you can start in the middle of the grid containing the symbol b), as long as you traverse every cell of the grid *at least once* whilst adhering to the cyclic order, possibly visiting some cells multiple times.

Not all mosaics are unlockable, however. There are some wherein there is no way to traverse every cell according to the rules given. Attempting to unlock these mosaics will instead lead to a curse of misfortune! Thus, your task is to determine whether a given grid of symbols is solvable given the conguration of the grid, the K symbols available, and the cyclic order they follow. For simplicity, we use the first K letters of the English alphabet as the symbols for this problem, with their cyclic order following the alphabetical order.

Input

The first line of input contains an integer T, the number of test cases in the file. T test cases follow.

The first line of each test case contains three space-separated integers R, C, and K. This is followed by R lines with C characters each, forming a grid representing a mosaic. These characters will be taken from the first K characters of the (lowercase) English alphabet.

Output

For each test case, output a line containing the string `FORTUNE!` if the given grid is solvable, otherwise output a line containing the string `MISFORTUNE!`.

Constraints

$1 \leq T \leq 100$
$1 \leq R, C \leq 30$
$2 \leq K \leq 26$

SAMSUNG Research

This problem is contributed by Samsung R&D Institute Philippines

Sample Input	Sample Output
3 3 3 3 abc cba abc 3 3 3 abc abc abc 3 4 3 abca acab cbac	FORTUNE! MISFORTUNE! FORTUNE!

Problem F
Party at the End of the Universe
Time Limit: 1.5 seconds

Billions of years in the future, you and the few remaining immortals sit on the barren wasteland that used to be Earth. All other civilizations have risen and fallen, and now the only thing left is for you and the others to sit back and wait for the final proton to decay. You turn to the other immortals and say, you know, all and all, we had a good life. Why don't we crack open some beers and watch the end of the universe?

You are planning an *end-of-the-universe watch party*. That's still a long ways away, but we might as well start planning now. There are n other immortals remaining, all of whom conveniently have single-letter names `A`, `B`, `C`, ... Also, the immortals are part of m different *group chats*, each of which contains some non-empty subset of the n immortals (can you imagine spending an eternity without the internet?)

There are k nights remaining until the end of the universe. You plan to do the following once every night, for each of those k nights:

- Select one of the group chats. Then, send a message to that group chat, inviting **everyone** there to your party. **Everyone** in that group chat is invited; you can't selectively only invite *some* people from that chat.

Note that there are m^k different ways to send these messages from now until the end of the universe.

Now, suppose you had a **must** list of immortals who *must* be invited to your party, and a **must-not** list that *must not* be invited to your party. For anyone in neither list, you *don't care* whether they are invited or not. Note that someone is invited if and only if you had sent a message to *any* of the group chats that they belong to.

Here's an interesting question—of the m^k different ways to send messages, how many of them result in everyone in the must list being invited, and everyone in the must-not list *not* being invited? This number can be quite huge, so output it modulo 998244341. And actually, you're not quite decided yet on how you feel about the other immortals, so you should answer T different test cases, each with a different must and must-not list.

Input

The first line of input contains three space-separated integers n, m, k.

Descriptions of the m group chats follow. This is a single line containing m space-separated strings of uppercase letters, with each string corresponding to a group chat. Each letter in eah string corresponds to some immortal that is a member of that group chat.

Then, a line of input containing a single integer T.

Descriptions of the T test cases follow. Each is described by a line containing two space-separated strings of uppercase letters, corresponding to the must list and the must-not lists, respectively. Again, each letter in each string corresponds to some immortal that is a member of that list. An empty must list or must-not list is instead represented by an `!` exclamation point.

Output

Output T integers, each of which contains the answer for the corresponding test case. Again, the answers should be given modulo 998244341.

Constraints

$1 \leq n \leq 16$
$1 \leq m \leq 10^5$
$1 \leq k \leq 10^{18}$
$1 \leq T \leq 2 \times 10^5$
Only the first n uppercase English letters appear in each group chat, must list, and must-not list (unless some list is empty, in which case it is represented by `!`)
In the list for some group chat, each immortal appears at most once.
In each pair of must and must-not lists, each immortal appears at most once across both lists.

Sample Input	Sample Output
5 3 4 ACDE ABC BCD 4 ! ! AC ! ABCD E B CD	81 80 14 0

Explanation

- If we do not care about who is or isn't invited, then there are simply $3^4 = 81$ valid sequences.
- If we must invite `A` and `C`, then we can show that there is only one sequence of message-sends that *doesn't* achieve this goal—if we send all the messages to the third *group chat*. All other sequences would be valid. Thus, the answer is $3^4 - 1 = 80$.
- We must invite `A`, `B`, `C`, and `D`, but we must **not** invite `E`. Thus, we should never send a message to the first group chat. However, as long as we send at least one message to the second group chat and one message to the third group chat, the sequence is valid. Thus, the answer is $2^4 - 2 = 14$.
- Note that all group chats with `B` also include `C` or `D`. Thus, the task is impossible, and we output that there are 0 valid ways.

Problem G
Song of GluGlu

Time Limit: 1 second

The legendary beast GluGlu roams around the galaxies, blessing all those who encounter it with a song, its rich and deep melody reverberating outwards from its entire body. Inexplicable power is imparted to anyone immersed in its song. The Song of GluGlu is an endless composition, constituted by some order of (possibly repeated) words. In total, there are N unique words which can make up the Song of GluGlu.

The power imbued by the Song of GluGlu doesn't come from the words themselves, but from the transitions from one word to another. The amount of power contained in the transition between two words A and B is the absolute difference (in terms of position in the alphabet) between the last letter of A and the first letter of B. For example, for the sequence of words `abah iemu youtah`, the power it contains from its two transitions is 1 from `h-i`, plus 4 from `u-y`, totalling 5.

Different orders of the same words could generate different levels of power. For example, given the same three words in our previous example, we can rearrange the sequence to `iemu youtah abah` which contains 11 units of power instead. Note that sequences could have repeated words, for example `iemu iemu abah`.

Through the research efforts of the Intergalactic Federation, they have made a device that could record a sequence of up to M words. Now they only need to know how much power can resonate out of it.

Given N unique words which can make up the Song of GluGlu, what is the maximum power contained within a sequence of M words?

Input

There is a single test case per file. The first line of the input contains two integers N and M. The second line of the input contains N space-separated words. Each word is composed of only lowercase alphabet characters.

Output

Output a single line containing an integer, denoting the maximum power contained within a sequence of M word.

Constraints

$1 \leq N \leq 10^5$
$1 \leq M \leq 10^{15}$
The length of any word is at most 10.

Sample Input	Sample Output
3 3 iemu youtah abah	37

Sample Input	Sample Output
3 6 alab yaz xoog	121

Explanation

For the first test case, the optimal word sequence is `iemu abah youtah`. The first transition generates 20 power, while the second transition generates 17, giving a total of 37.

Problem H
Specific Gravity
Time Limit: 2 seconds

Sarabar L is an alien on an expedition to a High Gravity planet. On this planet lives an intelligent native species and this species has developed technology to detect gravitational waves.

Sarabar's ship uses some fancy devices to make it so that the ship itself will be detected as having 0 weight and 0 volume (Newton hates her!). But it sure would be suspicious for a 0-density ship-like blob to be gliding through the atmosphere, right? To avoid detection, she decides that she must *increase* the density of her ship by picking up rocks from the planet so that her density ends up matching the atmospheric density of the planet. The density of the planet is D, and very curiously, it's actually an integer.

This planet has N different *types* of rocks. There are virtually infinite copies of each rock, and each rock has a certain weight and volume. Sarabar can pick up any finite number of rocks, and the resulting density of her ship will become the total weight of all the rocks that she picked up, divided by the total volume of the rocks that she picked up. Again, Sarabar wants her ship's density to be exactly equal to D.

Is there a way off this planet without getting detected by the gravitational wave sensors?

Input

The first line of the input contains a single integer T, the number of test cases. T test cases follow. Each test case is described in the following manner.

The first line of each test case contains two integers N and D, the number of different types of rocks and the density of the planet, respectively.

N lines follow, each containing two integers w_i and v_i denoting the weight and volume of each type of rock. You may assume that all rocks of the same type have the same weight and volume.

Output

For each test case, output a single line containing the string `BLAST OFF` if it's possible to leave the planet, and otherwise `CURSE YOU NEWTON` if it's not possible.

Constraints

$1 \leq T \leq 1000$
$1 \leq N \leq 5000$
$1 \leq w_i, v_i, D \leq 10^9$

Sample Input	Sample Output
3 2 3 2 1 4 1 4 7 1 7 8 2 140 3 15 2 3 3 10 3 4 1 8 2	BLAST OFF BLAST OFF CURSE YOU NEWTON

Explanation

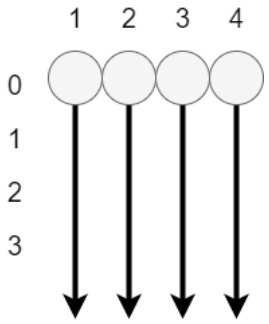
For the first test case, we have two types of rocks. We can achieve density 3 by getting one of each type of rock, getting total weight $2+4=6$ and total volume $1+1=2$ which results in density $6/2=3$.

Problem I
Teleporter Simulation

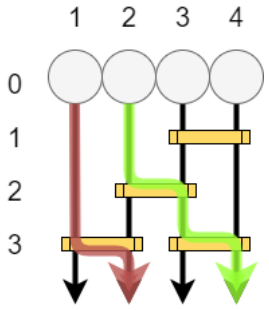
Time Limit: 2 seconds

In order to cut costs, the Intergalactic Federation outsources some of its non-critical work to 3rd-galaxy planets – planets with significantly low cost of living and, therefore, lower wages. One such planet is Earth of the Sol system, and you, as a developer of Earth, have been contracted by the Federation to help maintain their buggy software.

One of the pieces of software you're tasked to maintain is a teleporter routing system. There are N teleporters, positioned at $(1,0)$, $(2,0)$, and so on until $(N,0)$. Initially, all the teleporters are simply routed directly downwards until row 10^9, as shown below.



Along the way, horizontal teleporter bridges can connect two adjacent columns X and $X+1$ at some row Y. When teleporters pass through these bridges, they *must* move horizontally along these bridges, whether they start from the left or the right of the bridge. The following shows an example with bridges, as well as the new routes when these are in place.



Initially, there are no bridges. As modifications are done on the system, bridges are added or removed. Your task is to simulate these, and at some points, management will ask you to answer the question: With the current bridges in place, at which column will teleporter X pass through when it is at row Y, and at this point of its journey, how many bridges will it have passed? If there is a bridge at that position, then you need to return the column *after* it crosses the bridge and the bridge count should include this bridge.

Input

There is a single test case per input file. The first line contains N and M, the number of teleporters and the number of commands you need to process.

M lines follow. Each will follow one of the following formats:

1. `ADD X Y` – this adds a bridge that connects column X and $X+1$ at row Y.
2. `REM X Y` – this removes the bridge that connects column X and $X+1$ at row Y. It is guaranteed that this bridge exists.

3. `QUERY T Y` – requests a query for the position and bridge count of teleporter T at row Y.

Note that initially there are no bridges.

Output

For each query, output a line containing two space-separated integers X and C, the column and bridge count of the teleporter respectively.

Constraints

$2 \leq N \leq 20$
$0 \leq M \leq 10^4$
At any point, no two bridges will overlap. That is, if a bridge is added with the left end at X at row Y, then it is guaranteed that there is no bridge with left end at $X-1$, X, or $X+1$ at row Y.

Sample Input	Sample Output
4 11 QUERY 1 2 QUERY 3 1000000000 ADD 3 1 ADD 2 2 ADD 1 3 ADD 3 3 QUERY 2 1 QUERY 2 3 QUERY 4 1 REM 3 3 QUERY 2 3	1 0 3 0 2 0 4 2 3 1 3 1

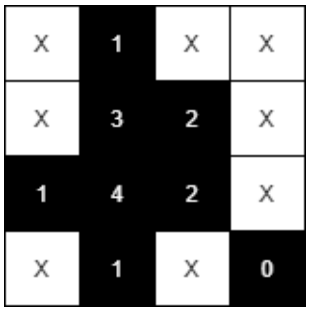
Problem J

Vintage Enephtys

Time Limit: 2 seconds

In the Intergalactic Federation's research on Enephtys, they have encountered Enephtys that don't adhere to the color of Red, Blue, Yellow. Instead, these so-called *Vintage Enephtys* are simply are black and white. There is even a certain niche market to these Enephtys, and the values could grow tremendously.

One of the Federation's scientists have proposed that the the value is somehow related to something they call **connection indices** of the pixels. The connection index of a black pixel is the number of neighboring pixels that are also black, in the up, down, left, and right directions. White pixels do not have a connection index. One such Enephty is shown below, and their connection indices are noted.



We count the number of pixels with connection indices $0, 1, 2, 3,$ and 4, and denote their counts as $C_0, C_1, C_2, C_3,$ and C_4. The Federation is convinced that Vintage Enephyts with certain combinations of these counts are valued higher.

The Federation is now ready to enter the Enephty market! They plan on making $N \times N$ Enephtys that have certain $C_0, \ldots, C_4$. Given N and C_0 through C_4, how many unique Enephtys can they make?

Input

The first line of the input contains an integer T, the number of test cases. T test cases follow.

Each test case is composed of two lines:

- A line containing an integer N, the length and width of the Enephtys the Federation plans on making.
- A line containing 5 space-separated integers C_0, C_1, C_2, C_3, C_4 denoting the counts of pixels in the Enephty that has connection indices $0, 1, 2, 3, 4$, respecively.

Output

For each test case, output a single line containing the number of possible Vintage Enephtys given the requirements.

Constraints

$1 \leq T \leq 500$
$1 \leq N \leq 4$
$0 \leq C_0 + C_1 + C_2 + C_3 + C_4 \leq N^2$

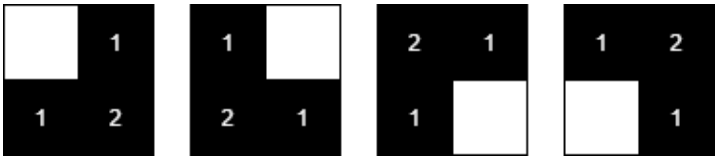
SAMSUNG Research

This problem is contributed by Samsung R&D Institute Philippines

Sample Input	Sample Output
4 2 0 2 1 0 0 2 2 0 0 0 0 2 3 0 0 0 0 4 2 4 0 0 1	4 2 0 32

Explanation

For the first test case, there are four possible Enephtys, as shown below.



Problem K
War of Human Conquest

Time Limit: 2 seconds

When the aliens first discovered humanity, they found us cute. So rather than hurling a large asteroid at our planet and wiping out all of humanity, they decided to subjugate our people instead and keep us as pets. As one of their offensive operations in the War of Human Conquest, they decided to bomb two factories from orbit.

The two factories, Factory A and Factory X, have N employees. Every employee works at both factories because a single minimum wage job wasn't enough to support them.

Because every employee has two jobs, they each have two shifts – one shift each for Factory A and Factory X. The shift at Factory A is identified by two numbers, A_i and B_i, the start and end times of the shift. The shift at Factory X is identified by X_i and Y_i, also the start and end times of the shift. These two shifts don't overlap. The numbers are also accurate to the millisecond because of late-stage capitalism employee surveillance policies.

All employees that are present at a factory when it is bombed will die. But everyone else still alive will still attend their shifts at the other factory until it is also bombed. The employees' meager livelihoods don't allow them to skip their shifts even if there is a direct threat to their lives.

As an alien sympathizer, your job is to find the best times to bomb Factory A and Factory X in order to maximize the total number of casualties.

Input

The input file contains a single test case. The first line contains a single integer N, the number of employees.

N lines follow, each containing four integers A_i, B_i, X_i, and Y_i describing the inclusive start and end times of the i th employees shifts in Factory A and X, respectively. These timestamps are represented by an integer, which is the number of milliseconds from midnight.

Note that it's possible that a shift covers past midnight, indicated by cases where $A_i > B_i$ or $X_i > Y_i$.

Output

Output a single integer K, the maximum number of employees that can be killed by nuking both factories.

Constraints

$1 \leq N \leq 10^5$
$0 \leq A_i, B_i, X_i, Y_i < 86400000$
The two shifts do not overlap.

This problem is sponsored by Expedock

Sample Input	Sample Output
5 59400000 64800000 21000000 29400000 5400000 10800000 21600000 28800000 75600000 1800000 35100000 36000000 8460000 31620000 7200000 8400000 48600000 52800000 8100000 11400000	3

Sample Input	Sample Output
7 0 1000000 2000000 3000000 0 1000000 2000000 3000000 0 1000000 2000000 3000000 0 1000000 4000000 5000000 4000000 5000000 6000000 7000000 4000000 5000000 8000000 9000000 4000000 5000000 10000000 11000000	6

Problem L
Yummy Tea

Time Limit: 1 seconds

The human race is close to extinction. When the Milky Way Galaxy collapsed, the few that remained fled to different points of the universe. You are one of the last humans, and are in the process of acclimating to your new home planet Betelgeuse V, and looking for a job.

Finally, you've decided to bring some cuisines of your old home planet Earth to your new neighborhood.

This has turned into something a bit difficult though, because Betelgeuse V doesn't necessarily have the same ingredients. After going through recipe after recipe, finally you've found that the ingredients of Milk Tea can be found within the shops of your small neighborhood! Thus, Yummy Tea is born.

Now you only need a place for you to set up your new venture. You've mapped out your neighborhood into an $N \times N$ grid, and noted some key locations:

- **M** is a milk shop. You're not sure what animal (or nut) they've milked it from.
- **B** is a tapioca pearls shop. When you call them "boba", they gape at you as if you've eaten their children.
- **T** is a "tea" leaf shop. The leaf is intoxicating to humans, but the Betelgeusians take it just fine.
- **S** is a sugar shop. As you've learnt, no semi-intelligent civilization is without their sugar.
- **P** is a disposable cup shop. You need something to put the drink in... right?
- **.** is *free real estate*, just empty lots waiting to be developed!

An example is shown below with $N = 4$.

```
BSB.
P...
TMMT
....
```

As it turns out, your samples are a pretty big hit. Through feedback, you've found out what Betelgeusians do and don't like with your recipe. Some don't want any sugar, and some don't like the tea aspect at all; just sugar and milk with tapioca pearls. Some don't want to even look at the tapioca pearls (probably those who heard you call it "boba"). An odd majority doesn't even want a cup, and carry their own reusable mugs. How weird!

From the feedback, you've tried different recipes by taking out some of the ingredients. Your new drinks can be represented as a string. For example, `MTP` is composed of only milk, tea, and the cup, and `MBS` has milk, *tapioca pearls*, and sugar (no cup!). Your old recipe can be represented as `MBTSP`. Note that the order of the letters do not matter.

In total, you have Q recipes, and for each of them you want to find the optimal location for you to set up shop that minimizes the sum of Manhattan distances to the nearest ingredients shops that you need to supply from. For example, if you have the recipe `MBTSP`, then the optimal location is represented below by the $\star$ character with total distance sum of $1+2+2+1+1=7$. On the other hand if you have the recipe **BM**, then the optimal location is represented below by the `#` character with total distance sum of $1+1=2$. Please note that you can only set up shop on *free real estate*.

```
BSB.
P*#.
TMMT
....
```

Given the Q recipes and the $N \times N$ grid of your neighborhood, what is the total distance sum of the optimal location of each recipe?

Input

There is a single test case per file. The first line of the input contains two space-separated integers N and Q. N corresponds to the size of the neighborhood grid, and Q represents the number of recipes you will need to check.

N lines follow, each containing a string of length N, corresponding to the neighborhood grid map. Each cell will always be one of the characters from `.MBTSP`.

A blank line follows, then Q lines, each containing a string corresponding to a recipe. Each recipe string has unique letters and consists of letters from `MBTSP`.

Output

For each of the Q recipes output a line containing the sum of distances to all the nearest required ingredient shops, assuming you chose the optimal location for that recipe. In total, you must output Q lines.

Constraints

$1 \leq N \leq 10^3$
$1 \leq Q \leq 10^4$
It is guaranteed that you can always find a location to set up shop in.

Sample Input	Sample Output
4 3 BSB. P... TMMT .... MBTSP T BM	7 1 2

Explanation

There are 3 recipes to check: `MBTSP`, `T`, and `BM`. The optimal locations are illustrated below, with the following legend.

- `MBTSP`: The optimal location is denoted by `*`. The total of the distances is $1+2+2+1+1=7$, respectively.
- `T`: One of the optimal locations is denoted by `@`. The total of the distances is 1.
- `BM`: The optimal location is denoted by `#`. The total of the distances is $1+1=2$, respectively.

```
BSB.
P*#.
TMMT
...@
```

Note that in reality, the optimal location may be the same for different recipes, and that's okay.