UP ACM

Algolympics 2016

Contents

Problem A
Gag Olympics
Time Limit: 2 seconds

GAGOLYMPICS is a relatively new contest wherein participants compete to deliver the best gag in order to become that year's Outstanding Gagmaster (OGag for short). For some reason, the losers of this competition like to call the year's Outstanding Gagmaster by his title. They especially find it amusing whenever they do not immediately hear and they'd have to keep on calling them.

The crew went ahead and did twenty sets of letter cutouts of the letters in `GAGOLYMPICS`, with the intention of posting them around the campus to publicize the competition.

Here is how their cutouts look like:

```
.###.   ..#..   .###.   .###.   #....   #...#   #.....#   ####.   #   .###.   .###.
#...#   .#.#.   #...#   #...#   #....   #...#   ##...##   #...#   #   #...#   #...#
#....   #...#   #....   #...#   #....   #...#   #.#.#.#   #...#   #   #....   #....
#.###   #####   #.###   #...#   #....   .###.   #..#..#   ####.   #   #....   .###.
#...#   #...#   #...#   #...#   #....   ..#..   #.....#   #....   #   #....   ....#
#...#   #...#   #...#   #...#   #....   ..#..   #.....#   #....   #   #...#   #...#
.###.   #...#   .###.   .###.   #####   ..#..   #.....#   #....   #   .###.   .###.
```

Their boss, who is not very imaginative, would like to know how the letters will look like when posted. And so, you are tasked to write a program that will output how the letters will look like when arranged to form different words.

Input

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

Each test case consists of a single line containing a single uppercase word.

Output

For each test case, output seven lines, representing what the sign will look like if the crew were to spell out the string indicated using their cutouts.

There should be a single blank column between letters, but no blank columns at the beginning or end. Use the character '`.`' (dot) for blanks.

Print a blank line after the output of each test case.

Constraints

$1 \leq T \leq 1000$
$1 \leq$ length of word ≤ 20
All letters of the word come from the letters of "GAGOLYMPICS"

Sample Input	Sample Output
3 ALL IS COOL	..#...#.....#.... .#.#..#.....#.... #...#.#.....#.... #####.#.....#.... #...#.#.....#.... #...#.#.....#.... #...#.#####.##### #..###. #.#...# #.#.... #..###. #.....# #.#...# #..###. .###...###...###..#.... #...#.#...#.#...#.#.... #.....#...#.#...#.#.... #.....#...#.#...#.#.... #.....#...#.#...#.#.... #...#.#...#.#...#.#.... .###...###...###..#####

Problem B
The Tale of the Extremely Bored Fairies
Time Limit: 2 seconds

This is a tale about extremely bored fairies. One may even choose to call it – a fairy tale!

We start our story talking about flowers. A hundred flowers are arranged in a linear path inside a magical garden. The first flower in the path is labelled 1, the second flower is labelled 2 and the third is labelled 4, etc. We'd tell you the labels of all the flowers, but let's just say that the i^{th} flower is labelled 2^{i-1}.

Today, n fairies visit the garden. The favorite numbers of each of the n fairies are $a+1, a+2, a+3, a+4, \ldots, a+n$. They think of their favorite number as they enter the garden.

Since they are such hipsters, they start from the 100^{th} flower in the garden, and visit the flowers in the reverse order. Like a `for` loop with `i--` as the third argument.

For each flower they visit, they look at the label. If the number they're thinking of is *greater than or equal to* the value on the label, they touch the flower. If they do touch the flower, they deduct the value of the label from the number they're thinking of. And so, they end up thinking of a number less than the number they were previously thinking of, unless they didn't touch the flower which means that they're still thinking of the same number they were thinking of before.

They repeat this process again and again until they finish visiting all 100 flowers in the garden. When this happens, they all fly away. As they fly away, they realize that they didn't do anything productive today.

Little did the fairies know that all the flowers that have been touched an odd number of times become paralyzed! Or as the Internet would say: They literally *can't even*!

That evening, an ogre went through the garden initially thinking about his favorite number, x. He then followed the same exact system the fairies did. Coincidentally, at the end of the ogre's trip to the garden, all the flowers were touched an even number of times that day. Hence, none of the flowers ended up being paralyzed, thanks to the ogre. You could say that the unexpected curse of the flowers was finally *ogre*.

And everyone lived happily ever after. Except the fairies. They're still bored. And so they ask you to figure out the ogre's favorite number so they can gossip about it.

Input

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

Each test case consists of a single line containing two integers, a and n, separated by a single space.

Output

For each test case, output a single line containing a single integer: the ogre's favorite number, x.

Constraints

$1 \leq T \leq 100000$
$0 \leq a \leq 10^{18}$
$1 \leq n \leq 10^{18}$

Sample Input	Sample Output
2 6 9 11 11	7 23

Problem C
Godlike Multiplication
Time Limit: 5 seconds

A team of DotA players tried to figure out how many Geomancer clones would be in the arena after using a Manta style.

In their modded version of DotA, a Geomancer always has X clones on the arena. Using a Manta style (from the original Geomancer's inventory) creates Y illusions for each Geomancer clone present on the arena.

They want to know how many illusions will be in the arena upon the activation of the Manta style. To solve this, they just need to multiply X and Y to get the answer. And they do this using normal grade school multiplication.

There is a problem though! Unlike regular DotA teams, none of the members of this team wants to *carry*. Thus, they do the grade school multiplication method without doing the carry operations.

For example, if there were 589 Geomancer clones and the Manta style makes 913 illusions for each clone, they would solve it like this:

```
    589
  x 913
 ------
   1547
   589
 4521
 ------
 458437
```

Given two numbers X and Y, find the result when multiplication is done this way.

Input

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

Each test case consists of a single line containing two positive integers: X and Y.

Output

For each test case, output a single line containing a single integer: the desired "product".

Constraints

$1 \leq T \leq 5$
$100 \leq X, Y < 10^{250000}$

Sample Input	Sample Output
4 589 913 913 589 1111 1111 1111111111 1111111111	458437 425437 1234321 1234567890987654321

Problem D
Colored Tile Puzzle
Time Limit: 10 seconds

Human! This problem is inspired by a computer game called Undertale.

You're gonna love this puzzle! You see these tiles? Once I throw this switch, they will begin to change color! Each color has a different function.

Red tiles are impassable! Hitting them will kill you. So don't. You also cannot walk on them. But I think death is a good enough motivation for you to not touch them. Yellow tiles are electric! They will electrocute you (and then send you back facing the opposite direction to the previous tile). Green tiles are alarm tiles! If you step on them, alarms will sound. And well, nothing much happens really. So I guess that's fine. Orange tiles are orange-scented. They will make you smell delicious! Blue tiles are water tiles. Swim through if you like, but if you smell like oranges, the piranhas will bite you (then you will die, so you shouldn't do that). Also, if a blue tile is next to a yellow tile, the water will also zap you, rendering the blue tile impassable (like a red tile). Violet tiles are slippery. You will slide to the next tile. However, the slippery soap smells like lemons which piranhas do not like! Passing through violet tiles will make you smell like lemons. Violets and blue are ok! Note that you can only smell like at most one fruit. You will smell like the most recent smell you've acquired. Finally, pink tiles. They don't do anything. Step on them all you like.

Understand? Oh. And you start at the very left of the $M \times N$ puzzle, not stepping on any of the tiles (not even on the puzzle area). You are free to choose which row you start with. Your first step will always be the one that leads you from outside the puzzle area to the first column of your chosen row (which will bring you to the first column, if this tile is accessible). Once you get inside the puzzle area, you cannot get out, unless you reach and are able to stop at a tile at the last column (the M^{th} column). Then, you take one last step to the right and voila! You have completed the puzzle.

You can also get out of the puzzle by going upwards from the first row or going downwards from the last row. Doing so will make you fall down a bottomless pit, so I don't recommend that. Nyeh heh heh!

What is the minimum number of steps in which the puzzle can be completed?

Input

Each test case will contain two space-separated integers N and M, where N is the number of rows and M is the number of columns in the rectangular grid.

The next N lines each contain a string of length M consisiting of only the following uppercase letters:

- `R` indicates a red tile at that position.
- `O` indicates an orange tile at that position.
- `Y` indicates a yellow tile at that position.
- `G` indicates a green tile at that position.
- `B` indicates a blue tile at that position.
- `V` indicates a violet tile at that position.
- `P` indicates a pink tile at that position.

Terminate the program if $M = N = 0$. Do not handle this case.

Output

For each test case, output a line containing the string "`Case X: Y`" where X is the case number and Y is the least number of steps needed to solve the puzzle. If it is unsolvable, take Y = "`LOSE`".

Constraints

$1 \leq M \leq 10^5$
$1 \leq N \leq 10^5$
$MN \leq 10^6$
$1 \leq$ number of test cases ≤ 23456

Warning: The input file is large (approximately 18Mb)! Please use fast I/O methods (e.g. `BufferedReader` in Java, `printf`/`scanf` in C/C++)

Sample Input	Sample Output
6 10 RRRRRRRRRR RRRRRRRRRR PPPPPPPPPP PPPPPPPPPP RRRRRRRRRR RRRRRRRRRR 6 10 BOOBVVVVVP RRRRRRRRRR BOOPVVBGBG RRRRRRRRRR BOOPVVVVVB RRRRRRRRRY 3 3 PPR RVR RVP 1 5 VPVPV 0 0	Case 1: 11 Case 2: 9 Case 3: LOSE Case 4: 3

Problem E
Noel
Time Limit: 2 seconds

Ghawckjebzmonfidvruqstpyx is very proud of his unique name. He likes it very much since it contains many distinct letters of the alphabet! Not only is it such a hassle to pronounce, it's also a hassle to type on this sheet. This is why we opted to use his nickname as this problem's title instead of the full version.

The protagonist of this story has made it a hobby to compare names. Given a pair of names, he would count the number of distinct letters in both names and declare the name with less distinct letters to be unoriginal. He's such a judgmental person!

Given a pair of names of people, figure out which one of them has a more unoriginal name. If they have the same number of distinct letters, output `EQUALLY UNORIGINAL` instead.

Input

The first line of input contains T, the number of test cases. Each test case consists of one line containing the two names, separated by a single space.

Output

For each test case, output a single line containing the answer.

Constraints

$1 \leq T \leq 5000$
$1 \leq$ length of name ≤ 55
Names consist of alphabet letters only, without spaces.

Sample Input	Sample Output
3 Hermione Granger Lando Calrissian Eiko Carol	Granger Lando Eiko

Explanation

In the first case, "`Granger`" has 5 distinct letters, which is fewer than the 7 distinct letters in "`Hermione`".

Problem F
Print F
Time Limit: 2 seconds

As of February 2016, the largest known prime number is $2^{74207281} - 1$, a number with 22338618 digits. It was found by the Great Internet Mersenne Prime Search (GIMPS).

As a consequence of this, the largest perfect number that we know so far is

$$2^{74207280} \times \left(2^{74207281} - 1\right).$$

(A *perfect number* is a number whose sum of proper divisors is equal to itself. A *proper divisor* is a divisor which is not equal to the number itself.)

Let F be the sum of the squares of the proper divisors of this perfect number. All we want is for you to print F. But this is too large. So we'll have to mod it out. See the input and output format for details.

Input

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

Each test case consists of a single line containing a single integer, M.

Output

For each test case, output a single line containing a single integer: the value of F modulo M, i.e. the remainder when F is divided by M.

Constraints

$1 \leq T \leq 20000$
$1 \leq M \leq 10^9$

Sample Input	Sample Output
2 1000 1001	626 1

Problem G
GGVV
Time Limit: 9 seconds

Emmanuel is a popular athlete in their school. Although uninvited, he sometimes joins his school's glee club and drama club. Unforunately, because of his busy schedule, he was only able to attend four sessions of biology classes this semester! Recently, he decided he wanted to run for student council. But his detractors are trying to prevent him from winning by pointing out that missing a lot of classes is not a good example to set to the student body. And so, he decides that he will study very hard for their upcoming biology exam to make up for his absences!

While in his all-nighter, a talking horse appears before Emmanuel. "Hello! I'm Vice Versa," says the horse. "Gandang gabi, Vice Versa," says Emmanuel. "Manny! I'm here to help you in your biology exam tomorrow! Let's study which animals can be observed to mate with the same sex!"

The talking horse gave him a sheet with something scribbles on it. Upon further inspection, you can see that it was a string of four different characters that resemble `!`, `i`, `P`, and `d`. Vice then tells Manny that the characters on the string gives instructions on where to see different species of animals which fall in love with the same sex.

Vice tells him to start from his house and face north. Look at the characters from left-to-right and follow these instructions one-by-one.

- If the character resembles `!`, move one step forward towards the direction you're facing.
- If the character resembles `i`, face towards the opposite direction you're currently facing and step forward towards the new direction you're facing.
- If the character resembles `d`, turn 90 degrees left and step forward towards the new direction you're facing.
- If the character resembles `P`, turn 90 degrees right and step forward towards the new direction you're facing.

After following these instructions, you will see a two animals that are of the same sex, which love each other (and mate with each other).

Manny, although reluctant, follows the instructions flawlessly and ends up in a place where two goats of the same sex are mating (because they love each other). Manny cannot believe his eyes! But his faith is strong! He still thinks that animals of the same sex should not be together. This was part of the (according to him) irrefutable teachings of someone who spent most of his life with twelve other dudes!

And so, Vice Versa tells him to go home. Vice Versa tells Manny to revise the string he

has. Since Vice is a horse, he is not good at articulating himself and he gives the revision in a cryptic way "`TYPE` X Y". This means that:

- If `TYPE` = "`dP`", Manny should look at characters between the X^{th} and Y^{th} character (inclusive) and replace all `d`'s with `P`'s and vice-versa. He should then write the new string down and throw away the old string.
- If `TYPE` = "`i!`", Manny should look at characters between the X^{th} and Y^{th} character (inclusive) and replace all `i`'s with `!`'s and vice-versa. He should then write the new string down and throw away the old string.

Manny does the necessary revisions correctly. He then goes home, faces North again and follows the instructions on the new string he has just obtained. He is then led to see another pair of animals of the same sex loving each other! Though a bit shaken by his new realization, Manny still sticks to his faith. Because of his stubornness, Vice gives Manny another set of set of revisions of the form "`TYPE` X Y" to lead him into another pair of animals.

After receiving a total of D revisions from Vice, he was able to expand his mind and said sorry to Vice, even though his sponsorship deal with a big company was discontinued. He told Vice that he should be able to love any horse of his choosing regardless of the sex.

Do you have a friend who was like Manny before he met Vice? Try listing down all the places where you can find these same sex animals loving each other, for future reference.

Input

The first line of input contains two integers separated by a single space, N and D.

The next line contains a string of length N consisting of characters from "`dPi!`", describing the initial set of instructions Manny received.

The next D lines represent the revisions made by Vice caused by Manny's stubbornness. Each line contains a string s (of length 2) and two integers X and Y. s is the `TYPE` and has either a value of `dP` or `i!`. The integers X and Y are as described in the problem statement.

Output

Output one line containing e_0 and n_0, describing the location of the goats. This means that the goats were e_0 steps east and n_0 steps north from Manny's house. Negative values for e_0 and n_0 mean to step west and south, respectively.

Output D more lines. The i^{th} line of these D lines should contain two integers, e_i and n_i, describing the location of the pair of animals found after the i^{th} revision, as in the case of the goats.

Constraints

$1 \leq N, D \leq 300000$
$1 \leq X \leq Y \leq N$
TYPE is either `dP` or `i`!

Warning: The input file is large (approximately 5Mb)! Please use fast I/O methods (e.g. `BufferedReader` in Java, `printf`/`scanf` in C/C++)

Sample Input	Sample Output
5 4 id!!P dP 1 5 dP 1 4 i! 1 3 dP 3 4	3 -2 -3 -2 3 0 1 2 1 2

Problem H
Timbre
Time Limit: 4 seconds

A new revolutionary app is gaining popularity in mobile devices these days! It's called TIMBRE and it's going to change the way you hear different pitches!

Upon opening TIMBRE, the user will hear a pitch. If the pitch the user is looking for is lower than what was just heard, the user must swipe left. The app will then play a pitch which is lower than the pitch played before swiping. Similarly, if the target pitch is higher than what was just played, the user must swipe right and the app must then play an pitch higher than the previous one.

Keeping this in mind, we note for any pitch uploaded in the app, there is at most two other pitches associated to it – the *left pitch*, which is the pitch that plays after the user swipes left, and the *right pitch*, which is the pitch that players after the user swipes right.

Take note that it is possible that there is no associated left pitch or right pitch to an uploaded pitch. In this case, the app simply crashes.

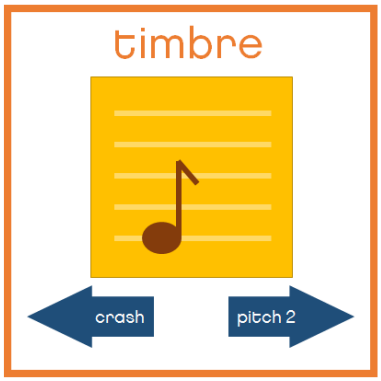
Now that the purpose of the app is clear, it's time to upload the pitches. The app is developed in such a way that you must upload the pitches one-by-one.

The first pitch uploaded will always be the first pitch played by the app.

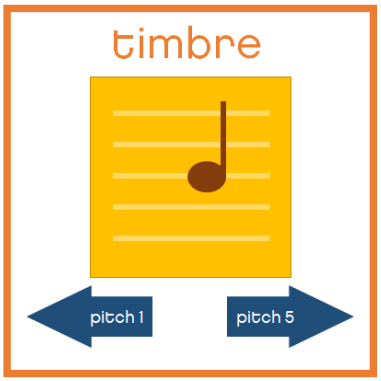
Upon uploading the subsequent pitches, it must then be saved as a *left pitch* or a *right pitch* of a previously uploaded pitch. Once it has been saved, it can not be overwritten.

For example, if the order of upload is $3, 1, 5, 4, 2$, the screenshots are illustrated on the next page.

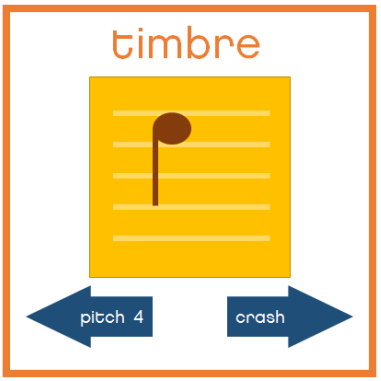
Given the upload order, can you tell which is the left pitch and right pitch for each uploaded pitch?



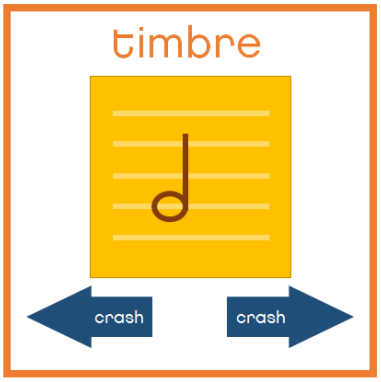
This is the screen for Pitch 1. You get here by swiping left from the first screen. The app crashes if you swipe left.



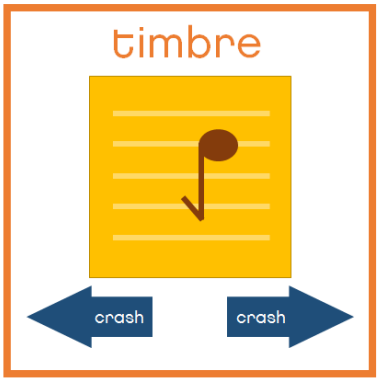
This is the screen for Pitch 3. It is the first screen you see upon loading the app.



This is the screen for Pitch 5. You get here by swiping right from the first screen. The app crashes if you swipe right.



This is the screen for Pitch 2. You get here by swiping right from Pitch 1. The app crashes if you swipe either way.



This is the screen for Pitch 4. You get here by swiping left from pitch 5. The app crashes if you swipe either way.

This is for the case when the upload order is `3 1 5 4 2`.

Input

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

The first line of each test case contains a single integer, N. The second line contains N integers $a_1, a_2, \ldots, a_N$, denoting the pitches of the notes in the database. The lower the value of the number, the lower the pitch.

Output

For each test case, output N lines where the i^{th} line is of the form

`L R`

where `L` is the pitch of the note that will be played if the user swipes left and `R` is the pitch

that will be played if the user swipes right from the pitch with value i. If the app will crash when the user swipes left (respectively right), write `L = 0` (respectively `R = 0`) instead.

Print a blank line after the output of each test case.

Constraints

$1 \leq T \leq 18$
$1 \leq N \leq 120000$
$[a_1, a_2, \ldots a_N]$ is a permutation of $[1, 2, \ldots, N]$

Warning: The input file is large (approximately 13Mb)! Please use fast I/O methods (e.g. `BufferedReader`/`StringBuilder` in Java, `scanf`/`printf` in C/C++)

Sample Input	Sample Output
3 3 1 2 3 3 2 1 3 5 3 1 5 4 2	0 2 0 3 0 0 0 0 1 3 0 0 0 2 0 0 1 5 0 0 4 0

Problem I
Inside Down and Upside Out
Time Limit: 10 seconds

"*Gravity's just a habit that you're pretty sure you can break.*" –OK Go

Sans has just constructed a room called the Gravitron. It's a room with dimensions (in meters): length L, width W and height H.

Sans stands *outside* the room, facing one of its faces (or walls). Let's label the walls of this box-shaped room.

1. We denote by **N** the face of the room which is *nearest* Sans, the one he's currently facing. This has dimensions $L \times H$.
2. We denote by **D** the face of the room which is the most *distant* to Sans, the wall opposite **N**. This also has dimensions $L \times H$.
3. We denote by **C** the *ceiling* of the room, which has dimensions $L \times W$.
4. We denote by **F** the *floor* of the room, which has dimensions $L \times W$.
5. We denote by **L** the wall to the *left* of Sans, which has dimensions $W \times H$.
6. We denote by **R** the wall to the *right* of Sans, which has dimensions $W \times H$.

Initially, the gravity in the room pulls objects towards **F**, the floor. However, Sans can direct the gravity to instead pull from **N**, **D**, **C**, **F**, **L** or **R** using hand gestures.

Cubes (of side length 1 meter) are placed (and stacked) inside the room such that when they are viewed from **C**, we can see a grid of $L \times W$ stacks of cubes of varying heights (possibly height 0, which means no cubes were at that "cell").

He does a sequence of gestures and eventually stops and returns the gravity to the floor, **F**. You may assume that switching the gravity is instantaneous and the gravity is very strong that it immediately pulls the cubes towards the appropriate direction.

Given the initial setup of the boxes, figure out the final setup of the boxes after Sans does his sequence of gestures.

Input

The input will contain several test cases.

The first line of each test case consists of three space-separated integers, L, W, and H, in that order.

The next line contains a string which consist exclusively of the characters `NDCFLR`. This describes the order of the gestures of Sans. Note that after this sequence of gestures, the gravity goes back to **F**.

The next W lines each contain L non-negative integers (whose values are at most H). These integers describe how high the stack of boxes are in each cell, when the boxes are viewed from the ceiling, with **L** and **R** on the left and right sides of the view, respectively.

To further clarify the orientation, the first integer of each line represents the boxes leaning against **L** and the last integer of each line represents the boxes leaning against **R**. Moreover, the first line of the W lines represents the boxes leaning against **D** and the last line represents the boxes leaning against **N**.

Terminate the program if $L = W = H = 0$. Do not process this test case.

Output

Output W lines, each containing L non-negative integers (whose value is at most H). This should describe how the stacks of cube end up after Sans does the sequence of gestures and after the gravity turns back into normal. Use the same orientation as described in the input.

Constraints

$1 \leq$ number of test cases ≤ 10000
$1 \leq L, W, H \leq 10$
$1 \leq$ length of string ≤ 500

Sample Input	Sample Output
3 2 2 N 0 1 0 0 0 0 2 3 2 L 0 0 0 1 0 0 3 2 2 D 0 0 0 0 1 0 2 3 2 R 0 0 1 0 0 0 4 3 5 NCLDFRLCNRFCD 0 0 0 0 0 0 1 0 0 0 0 0 2 3 4 LRFDCN 2 3 0 4 4 0 0 0 0	0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 3 0 4 2 4