

Problem A. The Alphabet Sticker

Program: sticker.(cpp|java)
Input: sticker.in
Balloon Color: Silver

When we were kids, we used to play with some stickers where these stickers contain some (but not necessarily all) lower case English alphabet letters.

Each sticker contains some letters arranged in a single row, where all occurrences of the same letter are adjacent to each other. A sticker can be represented as a string of characters, for example the following are valid stickers' representations: "aabcc", "ccccab" and "mmaw". And the following are not valid (because not all occurrences of the same letter are adjacent to each other): "abacc", "cccabc" and "mawm".

Now we found some stickers with some missing letters, but we are sure that all missing letters belong to the visible letters set (that is, for every missing letter, there is at least one visible letter that matches the missing one). In this problem a question mark letter represents a missing letter. Given some stickers' representations with zero or more missing letters, your task is to count the number of possible original configurations for each sticker.

For example, this sticker "aa??bb" with missing letters could have been one of the following original stickers "aaaabb", "aaabbb" or "aabbbb". But it could not have been any of the following original stickers "aababb" (it is invalid sticker) and "aacccb" (because the letter 'c' did not appear in the given configuration).

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each test case is described in one line which contains a non-empty string which consists of up to 10,000 letters, each letter is either a lower case English letter (from 'a' to 'z') or a question mark ('?'). This string represents a sticker configuration which contains zero or more question marks, it will also contain at least one letter which is not a question mark and there will be at least one valid original configuration for it.

Output

For each test case, print a single line which contains a single integer representing the number of possible original configurations for the sticker, since the result may be very large, print it modulo 1,000,000,007 ($10^9 + 7$).

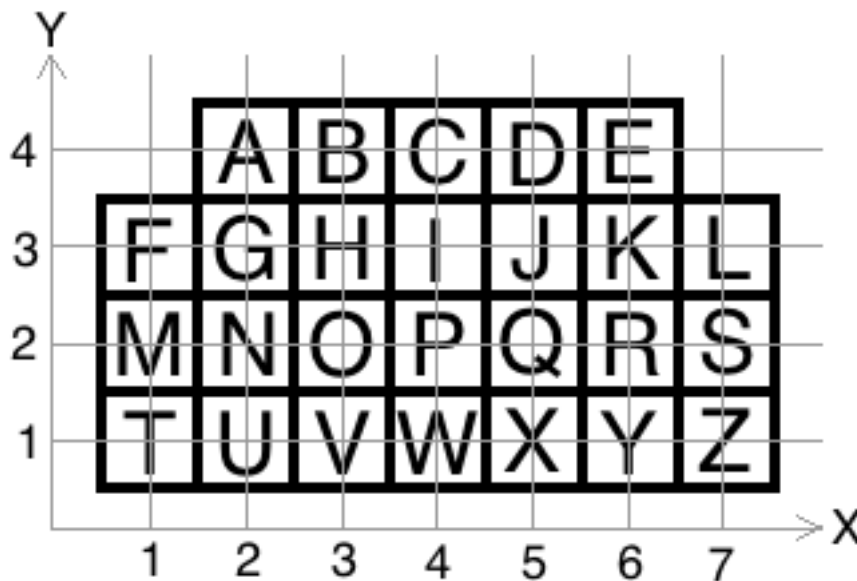
Examples

sticker.in	stdout
4	3
aa??bb	1
aaccbb	1
?a?	1
a??a	

Problem B. Swyper Keyboard

Program: keyboard.(cpp|java)
Input: keyboard.in
Balloon Color: Dark Pink

Swyper keyboard is a new kind of keyboards for touch screen mobile phones, where you just draw a sequence of one or more connected segments (poly-line) instead of tapping every letter of the word you want to type. The keyboard layout which we will deal with in this problem will look like the following one (the width and height of each square is 1 unit, and the center of each square is always a point with integer coordinates):



Please note that the keyboard which is described in this problem does not behave exactly like the real popular keyboard, so do not make any assumption which is not mentioned in the problem statement.

A poly-line in this problem will be represented using a string of 2 or more upper case English letters, where every 2 consecutive letters are always different. For example the string “ACM” represents a poly-line which starts from the center of the letter ‘A’ then goes to the center of the letter ‘C’ then goes to the center of the letter ‘M’. But this poly-line touches some other letters, between ‘A’ and ‘C’ it touches ‘B’ and between ‘C’ and ‘M’ it touches ‘B’ then ‘H’ then ‘G’ then ‘N’ (in that order). Given the initial string which represents a poly-line, we can get another string of all the letters which that poly-line touches in the correct order, which is referred to as the expansion of the poly-line. In this example this string is “ACBHGNM”. When you draw this poly-line (which was initially represented by the string “ACM”), you might mean any word which is a subsequence of the expansion string of the poly-line (this string is “ACBHGNM”), you might mean “ACM”, “BGN”, “ACBHGNM” or any other subsequence.

A string X is considered a subsequence of another string Y, when X can be obtained by deleting zero or more letters from the string Y without changing the order of the remaining letters.

You are given a dictionary containing one or more words and a string which is the initial representation for a poly-line, your task is to find a word in the dictionary which might be meant by that poly-line as described above.

Note that a poly-line which touches only the boundaries of a square does not count as a touch, for example a segment going from ‘A’ to ‘H’ does not touch ‘B’ or ‘G’.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, the first line of each test case contains an integer and a string separated by a single space $N P$ ($1 \leq N \leq 1,000$) representing the number of words in the dictionary and the initial representation of the poly-line, respectively. Followed by N lines, each line contains a single string which represents a word in the dictionary. All the strings in the input are non-empty strings of at least 2 and at most 100 upper case English letters (from 'A' to 'Z').

Note that the given string for the poly-line is just to describe its segments, and it might not include all the letters which the poly-line touches. Also, one poly-line might touch the same letter more than once, even though consecutive letters of a poly-line can not be the same.

Output

For each test case, print a single line which should be "NO SOLUTION" if there is no word in the dictionary which is a subsequence of the expansion of the given poly-line. Otherwise you should print the first word (according to the input's order) which is a subsequence of the expansion of the given poly-line.

Examples

keyboard.in	stdout
2	AM
3 ACM	NO SOLUTION
ACPC	
AM	
ACM	
2 ACM	
XY	
ZW	

Problem C. Increasing Shortest Path

Program: path.(cpp|java)
Input: path.in
Balloon Color: Yellow

We all love short and direct problems, it is easier to write, read and understand the problem statement. Here is one of these problems. *“Life is too short to make a story”, said Ahmed Aly.*

You are given a weighted directed graph of N nodes (the nodes are numbered from 1 to N), where the weights of the edges are distinct and positive. For each graph, you are also given a list of queries to answer.

Each query will be represented by 3 integers $A B C$, which means you need to find the shortest path (the path with minimum sum of weights of its edges) which goes from node A to node B and uses at most C edges, such that the weights of the edges in that path are in increasing order along the path, which means the weight of each edge in that path should be greater than the weight of the edge before it (unless it is the first edge in the path).

Your task is to write a program which answers these queries.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, the first line of each test case contains 3 integers separated by a single space $N M Q$ ($2 \leq N \leq 150$), ($0 \leq M \leq 3,000$) and ($1 \leq Q \leq 1,000$) representing the number of nodes, the number of edges and the number of queries, respectively. Followed by M lines, each line contains 3 integers separated by a single space $X Y Z$ ($1 \leq X, Y \leq N$) ($1 \leq Z \leq 3,000$) which represent an edge going from the node X to the node Y with cost Z (X and Y will be different). Followed by Q lines, each line contains 3 integers separated by a single space $A B C$ ($1 \leq A, B \leq N$) ($0 \leq C \leq M$) which represent a query as described above (A and B will be different).

Note that there might multiple edges between the same pair of nodes.

Output

For each test case, print a single line for each query which contains a single integer, the minimum sum of weights for a path between the given pair of nodes which satisfies the given constraints, or -1 if there is no valid path between the given nodes which satisfies the given constraints. The output must not contain empty lines between the cases.

Examples

path.in	stdout
1	17
8 9 3	6
1 2 1	-1
2 3 2	
3 4 3	
4 5 12	
5 8 7	
1 6 8	
6 4 9	
1 7 5	
7 4 4	
1 4 2	
1 4 3	
1 4 1	

Problem D. Cup of Cowards

Program: `monster.(cpp|java)`
Input: `monster.in`
Balloon Color: `Orange`

Cup of Cowards (CoC) is a role playing game that has 5 different characters (Mage, Tank, Fighter, Assassin and Marksman). A team consists of 5 players (one from each kind) and the goal is to kill a monster with L life points. The monster dies if the total damage it gets is at least L . Each character has a certain number of allowed hits, each hit has a certain damage and a certain cost (the cost and damage might be different for each character). The team wants to kill the monster using the minimum cost so they can perform better in later missions. They want your help to find the minimum cost they will pay to kill the monster and the damage they should incur on it.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, the first line of each test case contains 1 integer L ($0 \leq L \leq 10^{12}$) representing the life points of the monster. Followed by 5 lines, each one contains 3 integers separated by a single space $H D C$ representing the maximum number of hits, the damage by each hit and the cost of each hit by one of the characters, respectively ($0 \leq H \leq 1,000$), ($0 \leq D, C \leq 10^9$) and the sum of the maximum number of hits for all characters will not be more than 1,000.

Output

For each test case, print a single line which contains 2 space separated integers, the first is the minimum cost for the hits used to kill the monster and the second is the damage incurred upon the monster. If there is more than one way to kill the monster using the same minimum cost, select the one with the least damage and if there is no way to kill the monster print "We are doomed!!"(without the quotes).

Examples

monster.in	stdout
2	19 33
33	We are doomed!!
2 3 4	
3 1 2	
4 3 2	
1 7 1	
3 4 2	
51	
3 3 1	
4 3 2	
2 3 3	
3 1 4	
5 2 3	

Problem E. Balloons Colors

Program: balloons.(cpp|java)
Input: balloons.in
Balloon Color: Green

Assigning a balloon color to each problem is one of the tasks we need to do every year, and sometimes it is tricky.

We noticed that some contestants assume some colors for some problems according to the difficulty. For example, the easiest problem is the red one and the hardest problem is the black one.

We do not want these assumptions to be true, so we decided to add constraints for the easiest and the hardest problems.

There are N problems, numbered from 1 to N , the easiest problem is problem number 1, and the hardest problem is problem number N . Also there are N unique colors, for simplicity we will give each color a unique number from 1 to N .

We want to assign each color to exactly 1 problem, such that the easiest problem does not get the color X and the hardest problem does not get the color Y .

Given N , X , Y and an assignment of the colors, your task is to find if this assignment satisfies the above conditions or not.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, the first line of each test case contains 3 integers separated by a single space $N X Y$ ($3 \leq N \leq 100$) and ($1 \leq X, Y \leq N$) representing the number of problems, the color which the easiest problem should not get and the color which the hardest problem should not get, respectively. Followed by a line which contains N integers separated by a single space (each integer from 1 to N should appear exactly once), the first integer is the color for the first problem (the easiest), the second integer is the color for the second problem and so on (the last integer is the color for the hardest problem).

Output

For each test case, print a single line which contains a single word, this word should be (without the quotes):

- "BOTH": If both the easiest and hardest problems got colors which they should not get.
- "EASY": If only the easiest problem got a color which it should not get.
- "HARD": If only the hardest problem got a color which it should not get.
- "OKAY": If both the easiest and hardest problems got colors which they can get.

Examples

balloons.in	stdout
4	BOTH
3 1 2	EASY
1 3 2	HARD
5 3 4	OKAY
3 1 2 4 5	
6 1 6	
2 1 3 4 5 6	
7 7 7	
1 7 2 3 4 5 6	

Problem F. NASSA's Robot

Program: `robot.(cpp|java)`
Input: `robot.in`
Balloon Color: Red

NASSA's robot landed on Mars. The place where it landed can be modeled as an infinite 2-dimensional plane with perpendicular X-axis and Y-axis coordinates.

The robot continuously reports its location back to Earth, but due to a serious design flaw, it only reports the moves it makes instead of the coordinates of its exact location. Some signals went missing and never reached our reception.

In one of the space exploration missions, the robot sent a sequence of signals, which can be represented by a string composed of the following characters: 'U', 'R', 'D', 'L' or '?'. 'U' represents up (Y-coordinate increases by 1), 'R' represents right (X-coordinate increases by 1), 'D' represents down (Y-coordinate decreases by 1), 'L' represents left (X-coordinate decreases by 1) and '?' represents a missed signal. Every character in the sequence is a single step in the corresponding direction. A missed signal is a single step in one of the four directions. The robot is initially at X-coordinate 0 and Y-coordinate 0 before starting to send the given signals.

After sending some signals while the robot is moving, its software crashed and the robot could not do any further moves. The researchers on the base want to limit the space where they can look for the robot. In other words, they want to find the minimum possible X-coordinate, the minimum possible Y-coordinate, the maximum possible X-coordinate and the maximum possible Y-coordinate of the current location of the robot.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each test case is described in one line which contains a non-empty string which consists of up to 100,000 letters, each letter is 'U', 'R', 'D', 'L' or '?'. This string represents the sequence of signals as described above.

Output

For each test case, print a single line which contains 4 integers separated by a single space, which are the minimum possible X-coordinate, the minimum possible Y-coordinate, the maximum possible X-coordinate and the maximum possible Y-coordinate for the location of the robot after it stopped moving.

Examples

<code>robot.in</code>	<code>stdout</code>
3	-1 -2 3 2
RUL?R?D	-8 -8 8 8
?????????	3 2 3 2
RRRUU	

Problem G. The Stones Game

Program: stones.(cpp|java)
Input: stones.in
Balloon Color: Blue

The stones game is a simple game, it is also a very old game which is unknown to almost everyone.

The game starts with N stones and M players, the players are numbered from 1 to M . The players play in turns, player number 1 plays first, then player number 2 and so on until player number M plays, after this player number 1 plays again and they keep playing until the end of the game.

For each turn, the players do the following 2 steps:

1. The player gets a chance to remove a stone, and he/she should remove a stone in this step if he/she decided to do so.
2. Regardless of the decision of the current player (whether or not he/she removed a stone in the first step), if this is not the first turn and in the previous turn the player decided not to remove a stone in his/her first step, then the current player must remove a stone in this step (if in the previous turn the player decided to remove a stone in his/her first step, then the current player must not remove a stone in this step).

This means in some turns a player might remove 0, 1 or 2 stones according to the above rules. In this game, the player who removes the last stone wins the game.

Now you are given the total number of stones, the total number of players and a player number and you are asked to answer the following question:

Is there a strategy for this player to win the game regardless of the actions taken by the other players in their turns?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each test case is described in one line which contains 3 integers separated by a single space $N M X$ ($1 \leq N, M \leq 10^9$) and ($1 \leq X \leq M$) representing the number of stones, the number of players and the player number, respectively.

Output

For each test case, print a single line which contains a single word, this word is either "YES" or "NO" (without the quotes) representing the answer for the above question for the given player number.

Examples

stones.in	stdout
2	YES
2 2 2	NO
2 2 1	

Problem H. Super Ants


Program: ants.(cpp|java)
 Input: ants.in
 Balloon Color: White

Here is another game, this game is called the ants game. In this game, you are given a grid, with N rows and M columns. The rows are numbered from 1 to N from top to bottom, and the columns are numbered from 1 to M from left to right. Each cell in the grid has unlimited store of a certain type of sugar, each unit of sugar gives the player a specific score. The game is just 1 step, given 1 super ant, position it in any cell in the grid, and the ant will do its magical work to determine your score.

Once a super ant is placed in one of the cells, it does a systematic behavior to gather sugar units from the stores. If there is no remaining time, the ant gets 1 unit of sugar from the store of its cell and stops working. If there is some remaining time, it starts a magical operation, which is the ants cloning operation. The super ant starts to clone itself to all the possible cells according to the current remaining time (all of its cloning operations start at the same time). It takes D seconds to clone itself to a cell that is far with D units. An ant at position (R_1, C_1) is far from the position (R_2, C_2) by $\max(|R_1 - R_2|, |C_1 - C_2|)$ units (where $|X|$ means the absolute value of X). Once a new super ant is cloned, it behaves as a super ant by utilizing the remaining time.

At the end, when the remaining time is not enough for any ant to clone itself to any other cell, each ant will collect one unit of sugar from its cell and your score is the sum of the values of all the collected sugar units. Note that an ant can not clone itself to another cell with distance more than the current remaining time, and an ant can not clone itself to the same cell more than once, and it can not clone itself to its cell, and any cell can contain any number of ants at any time.

Okay, which positions an ant can clone itself to them? First of all, an ant can not clone itself to a cell outside of the grid. Second, it can only clone to any cell in one of its 8 directions' vectors. Direction vector means all consecutive cells on the same direction, for example from position (A, B) , east vector will generate $(A, B + 1)$, $(A, B + 2)$, $(A, B + 3)$ and so on. In the below image, a super ant with 2 seconds remaining, can clone itself to 16 positions, 8 of them with distance 1 and 8 with distance 2.

	2		2		2	
		1	1	1		
	2	1		1	2	
		1	1	1		
	2		2		2	

Given the score value of each sugar unit in the grid, the total allowed time you have (the time starts once the first ant is placed) and a cell which the first super ant will start from it, can you write a program which calculates the score you will get if you used that cell?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each test case starts with a

line containing 3 integers separated by a single space $N M S$ ($1 \leq N, M \leq 50$) and ($0 \leq S \leq 500$) representing the number of rows in the grid, the number of columns and the remaining time, respectively. Followed by a line containing 2 integers separated by a single space $I J$ ($1 \leq I \leq N$) and ($1 \leq J \leq M$) representing the row number and the column number of the cell which the first ant will start from it, respectively. Followed by N lines each line contains M integers separated by a single space, representing the value of each sugar unit in each cell in that row. Each integer in the grid will not be less than 0 and will not be greater than 9.

Output

For each test case, print a single line which contains a single integer representing the score you will get at the end of the game, since the result may be very large, print it modulo 1,000,000,007 ($10^9 + 7$).

Examples

ants.in	stdout
2	45
5 6 1	349
3 3	
0 2 4 6 8 1	
1 1 2 3 1 2	
1 4 5 6 3 4	
2 7 8 9 5 8	
3 5 8 0 7 0	
5 6 2	
3 3	
0 2 4 6 8 1	
1 1 2 3 1 2	
1 4 5 6 3 4	
2 7 8 9 5 8	
3 5 8 0 7 0	

Note

In the first example, the first ant will collect one sugar unit from its cell with score 5. In addition, it will clone 8 ants in the 8 directions, each one of them can only get 1 sugar unit from its cell. The total score is $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 = 45$.

In the second example, the first ant will clone itself to 8 cells at distance 2, with remaining time 0, and to 8 cells at distance 1 with remaining time 1 (thus each of the latter clones will further clone itself).

Problem I. Omar Loves Candies

Program: `sum.(cpp|java)`
Input: `sum.in`
Balloon Color: `Black`

Omar loves to eat a lot of candies, but unfortunately most of the candies are not healthy. So his parents found a way to give each candy a score, a higher score means a healthier candy (the score is an integer that can be positive, zero or negative).

One day he went with his parents to buy some candies, and they found a strange store where all the candies are stored in a 2-dimensional grid of N rows with M candies in each row. The rows are numbered from 1 to N from top to bottom, and the columns are numbered from 1 to M from left to right and every cell contains one candy.

They noticed something else, any candy (except for those in the first row) is healthier than the candy which is exactly above it, and any candy (except for those in the first column) is healthier than the candy which is exactly to its left (healthier means having higher score as defined above).

There is one more strange thing about this store, to buy some candies you have to select a sub-rectangle of the candies' grid and buy all the candies within this sub-rectangle.

Omar's parents want to select a non-empty sub-rectangle that has the maximum sum of candies' scores among all possible sub-rectangles.

For example, consider the grid in the example input. Some of the possible sub-rectangles of candies they can select are $[-2, -1, 2, 3]$, $[-4, -2, -1]$ or $[2, 3, 4, 5]$. The last sub-rectangle has the maximum sum of scores, which is 14. They can not select the following lists of candies $[1, 2, 3, 4, 5]$ or $[-2, -1, 2]$ (because these lists do not form a sub-rectangle of the given grid).

-4	-2	-1
-3	2	3
1	4	5

Can you help them by writing a program which finds the non-empty sub-rectangle with the maximum possible sum of scores in the given grid?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each test case starts with a line containing two integers separated by a single space $N M$ ($1 \leq N, M \leq 1,000$) representing the dimensions of the candies' grid, followed by N lines, each one contains M integers separated by a single space, representing the candies' scores in this row. The given grid representation will satisfy the conditions mentioned above, and each integer in the grid will not be less than -2,000 and will not be greater than 2,000.

Output

For each test case, print a single line which contains a single integer representing the maximum sum of scores they can get from a non-empty sub-rectangle.

Examples

sum.in	stdout
1 3 3 -4 -2 -1 -3 2 3 1 4 5	14

Problem J. Modified LCS

Program: sequence.(cpp|java)
Input: sequence.in
Balloon Color: Purple

LCS stands for longest common subsequence, and it is a well known problem. A sequence in this problem means a list of integers, and a sequence X is considered a subsequence of another sequence Y , when the sequence X can be obtained by deleting zero or more elements from the sequence Y without changing the order of the remaining elements.

In this problem you are given two sequences and your task is to find the length of the longest sequence which is a subsequence of both the given sequences.

You are not given the sequences themselves. For each sequence you are given three integers N , F and D , where N is the length of the sequence, F is the first element in the sequence. Each element except the first element is greater than the element before it by D .

For example $N = 5$, $F = 3$ and $D = 4$ represents the following sequence: [3, 7, 11, 15, 19].

There will be at least one integer which belongs to both sequences and it is not greater than 1,000,000.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each test case is described in one line which contains 6 integers separated by a single space $N1 F1 D1 N2 F2 D2$ ($1 \leq N1, N2 \leq 10^{18}$) and ($1 \leq F1, D1, F2, D2 \leq 10^9$) representing the length of the first sequence, the first element in the first sequence, the incremental value of the first sequence, the length of the second sequence, the first element in the second sequence and the incremental value of the second sequence, respectively.

Output

For each test case, print a single line which contains a single integer representing the length of the longest common subsequence between the given two sequences.

Examples

sequence.in	stdout
3	4
5 3 4 15 3 1	3
10 2 2 7 3 3	50
100 1 1 100 1 2	

Problem K. Mario Kart

Program: mario.(cpp|java)
Input: mario.in
Balloon Color: Light Pink

Have you ever played the Mario game? Of course you did, who did not?! Anyway, a new version of the Mario game has been released, it is some kind of kart racing game. And you decided to write a program to find the best strategy for you to complete each level.

Each level track can be modeled as an infinite straight line, with some stations at some specific points on this line. Each station has an integer, representing its position on the track. Your task is to go from the first station (the one with smallest position) to the last one (the one with largest position) in the minimum number of moves.

You can move between any two stations directly (you can go to a non-adjacent station, or you can go back to a station with a lower position if you want!) if you have enough boost coins for that move. In each level, you have some boost coins that you can use. Each boost coin has a cost and power value. You can select any subset of the coins for each move, but each coin may be used only once per move. Coins are permanent, and you can use the coins again for other moves in the same level.

To make a move, you must choose a subset of the boost coins, such that the sum of their costs must not exceed L , and the sum of their power values must be exactly equal to the absolute difference between the positions of the two stations you are moving between. If there is no such subset, then you cannot make that move directly.

Now you are given some configurations for some levels, and you are required to find the minimum number of moves to finish each one, or say it is impossible to finish that level.

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, the first line of each test case contains 3 integers separated by a single space $N M L$ ($2 \leq N \leq 100$), ($1 \leq M \leq 100$) and ($1 \leq L \leq 1,000$) representing the number of stations, the number of boost coins and the maximum sum of coins' costs in each move, respectively. Followed by a line which contains N unique positive integers separated by a single space (not necessary sorted, and each integer will be at most 1,000), representing the positions of the stations. Followed by M lines, each line contains 2 integers separated by a single space $C V$ ($1 \leq C, V \leq 100$) which represent the cost and the power value of a coin, respectively.

Output

For each test case, print a single line which contains a single integer, this integer should be -1 if there is no way to go from the left most station to the right most station, or the minimum number of moves to do it if it is possible.

Examples

mario.in	stdout
2	2
3 2 4	-1
3 1 6	
3 2	
3 3	
3 1 4	
1 3 6	
3 2	

Note

In the first test case, the station positions are [3, 1, 6], and you start at 1 and must end at 6. You will have to make 2 moves, going from 1 -> 3 using the coin (3, 2), and going from 3 -> 6 using the coin (3, 3). You can not go directly from 1 -> 6 using (3, 2) and (3, 3) because the sum of the costs of the coins exceeds the limit 4.

Problem L. Omar's Bug

Program: bug.(cpp|java)
Input: bug.in
Balloon Color: Gold

Omar is the youngest programmer in the world, he is just 11 months old. He has read about the binary search algorithm, and he decided to write it by himself.

He wrote the following function (its syntax is valid and it does the same job in both C++ and Java):

```
int findFirstGreaterThanOrEqual(int array[], int N, int X) {
    int start = 0, end = N;
    while (start < end) {
        int middle = (start + end) / 2;
        if (array[middle] > X) {
            end = middle;
        } else {
            start = middle + 1;
        }
    }
    return start;
}
```

The function will always be called with parameters which satisfy the following conditions:

1. The first parameter is an array which contains at least 1 integer and at most 99 integers.
2. The array contains distinct integers and it is sorted in increasing order.
3. The integers in the array are positive and each one is at most 100.
4. N is the number of elements in this array.
5. X is a positive integer which is at most 100.

Here is how Omar wants the function to work:

If there is no integer in the given array which is greater than or equal to X , Omar wants the function to return N . Otherwise, Omar wants the function to return the index of the smallest integer in the array that is greater than or equal to X (the first element in the array has index 0).

Unfortunately there is a bug in the function and it returns wrong result for some cases, and Omar can not find the bug (he is too young). Can you help him to generate some arrays to test his function?

Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T , the number of test cases ($1 \leq T \leq 100$). Followed by the test cases, each test case is described in one line which contains 3 integers separated by a single space $N X Y$ ($1 \leq N \leq 99$), ($1 \leq X \leq 100$) and ($1 \leq Y \leq 2$). N is the value of the second parameter and X is the value of the third parameter. If Y is 1, you should generate an array which when passed to the function with the given parameters, the function will return a correct result. If Y is 2, you should generate an array which when passed to the function with the given parameters, the function will return a wrong result. In both cases, the generated array must satisfy the above conditions.

Note that correct or wrong result is relative to the way which Omar wants the function to work.

Output

For each test case, print on a single line, N integers separated by a single space, the first integer is the first integer in the generated array, the second integer is the second integer in the generated array and so on (the generated array should satisfy the above conditions). If there is more than one correct solution, print the lexicographically smallest one.

When comparing two different arrays of the same length, the lexicographically smaller array is the one with a smaller value on the smallest index where they differ.

Examples

bug.in	stdout
2	1 2 4
3 3 1	1 2 3 7
4 7 2	