Preliminaries

for the 2013 Benelux Algorithm Programming Contest

The Problem Set

Almost blank page

A Area Coverage

In this day and age, a lot of the spying on other countries is done with the use of satellites and drones equipped with cameras. All these photographs of various sizes and from various sources can be combined to give a picture of the country as a whole.

Given the photographs (that is to say, the rectangular area covered by each, since the contents of the photographs themselves are of course top-secret!), can you work out what the total area is of all that is photographed? Note that certain areas can appear on multiple photographs and should be counted only once.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with an integer n ($1 \leq n \leq 1\,000$): the number of photographs.
- n lines with four space-separated integers x_1, y_1, x_2 and y_2 ($0 \leq x_1, y_1, x_2, y_2 \leq 1\,000\,000$, $x_1 < x_2$ and $y_1 < y_2$): the coordinates of the southwest and northeast corner, respectively, of each photograph. The photographs are all rectangular in shape with their other corners at (x_1, y_2) and (x_2, y_1).

The coordinates correspond to a flat two-dimensional space (i.e. we assume the Earth to be flat).

Output

Per test case:

- one line with an integer: the total area of all that appears on the photographs.

Sample in- and output

Input	Output
2 3 0 6 20 16 14 0 24 10 50 50 60 60 2 0 0 20 10 10 4 14 8	376 200

Almost blank page

B Bad Signal

There is an important UN meeting in town. Any self-respecting espionage agency will try to eavesdrop on the delegations to gain some advantage in the negotiations. They do this by planting hidden microphones in and around the meeting places. Those microphones continuously capture sound waves and transmit them via radio.

In fact, fierce competition between espionage agencies has left the whole city scattered with hidden microphones, so much so that the radio waves interfere with each other and it is often not even possible to make out any signal in the mess of radio waves – depending on your position and proximity to the different transmitters obviously.

Specifically, it is possible to make out a signal i if and only if:

$$r_i > 6(B + \sum_{j \neq i} r_j)$$

Where

- $r_i = \frac{s_i}{|P_i - P_{listen}|^2}$ is the strength of the received signal of microphone i,
- s_i is the strength of the signal sent from microphone i,
- P_i is the position of microphone i,
- P_{listen} is the position where you are listening to the signals,
- $|P_i - P_j|$ is the Euclidean distance between points P_i and P_j and
- B is the level of background noise.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with an integer n ($0 \leq n \leq 100\,000$): the number of planted microphones.
- one line with the integer B ($0 \leq B \leq 1\,000\,000$): the level of background noise.
- one line with two space-separated integers x and y: the x and y coordinates of the location P_{listen} where you receive the signals.
- n lines with three space-separated integers x_i, y_i and s_i ($0 < s_i \leq 1\,000\,000$): the x and y coordinates of the location P_i of microphone i and its signal strength, respectively.

All coordinates are in the range $[0; 10\,000]$. The locations P_i all differ from P_{listen}. The test data is constructed so that small floating point rounding errors will not influence the outcome of any solution.

Output

Per test case:

- one line with an integer: the (one-based) index of a microphone, the signal of which can be made out, or the string "`NOISE`" if there is no such microphone.

Sample in- and output

Input	Output
3 4 10 100 100 90 90 20000 110 90 50 90 110 1000 110 110 50 4 100 100 100 90 90 20000 110 90 50 90 110 1000 110 110 50 2 0 0 10 0 0 1000 0 8 1	1 NOISE 1

C Cracking the Safe

Secret agent Roger is trying to crack a safe containing evil Syrian chemical weapons. In order to crack the safe, Roger needs to insert a key into the safe. The key consists of four digits. Roger has received a list of possible keys from his informants that he needs to try out. Trying out the whole list will take too long, so Roger needs to find a way to reduce the list.

A valid key satisfies a certain condition, which we call the 24 condition. Four digits that satisfy the 24 condition can be manipulated using addition, subtraction, multiplication, division and parentheses, in such a way, that the end result equals 24.

For example, the key $(4, 7, 8, 8)$ satisfies the 24 condition, because $(7-8/8)*4 = 24$. The key $(1, 1, 2, 4)$ does not satisfy the 24 condition, nor does $(1, 1, 1, 1)$. These keys cannot possibly be the valid key and do not need to be tried.

Write a program that takes the list of possible keys and outputs for each key whether it satisfies the 24 condition or not.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with four space-separated integers a, b, c, d ($1 \leq a, b, c, d \leq 9$): a possible key.

Output

Per test case:

- one line with either "`YES`" or "`NO`", indicating whether the key satisfies the 24 condition or not.

Sample in- and output

Input	Output
4 4 7 8 8 1 1 2 4 1 1 1 1 1 3 4 6	YES NO NO YES

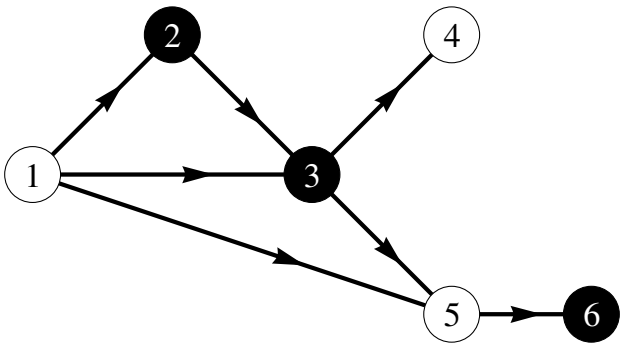
Almost blank page

D Dreaded Alternating Game

To determine their social position, spies like to play games of wit and cunning. One of these games is the dreaded alternating game. It is played by two players, who are called player even and player odd. These strange player names are used to determine the winner of each game: player even wins if the number of moves in the completed game was even, and similarly player odd wins if the number of moves was odd.

Every game is played on a specially prepared game board, consisting of places and one-way connections between certain pairs of places. Every place is controlled by one of the two players. The game starts by putting a token on the starting place of the game board. Every turn, the player who controls the place where the token resides must make a move by moving the token along one of the outgoing connections of that place. This continues until no more move is possible, at which point the winner of the game is known. All game boards are constructed such that infinite games are impossible.

The rules of the game state that both spies playing the game must come to an agreement on which player each one will be. The spies have figured out that picking their player is the most important part of the game. Special agent Walker wants to win every game she plays to improve her social position. She can play the game perfectly if she knows which player to pick at the start of the game. She developed a special set of skills to ensure that she will always get to be the player she wants to be. It is your task to help her pick the best player for a specific game board. Better make sure it is correct; you do not want to disappoint special agent Walker.



A visual representation of the third sample. The dark circles indicate places controlled by the even player. Despite having the first move, the odd player loses. If he moves the token from 1 to 3, the even player wins directly by moving the token to 4. If the odd player moves the token to 2, the even player first moves the token to 3 (as he must) and then to 5, after which the odd player is forced to move the token to 6 and thus loses (four moves were made). Similarly, moving the token to 5 at the first move leads to an automatic win for the even player.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with three space-separated integers n, c and s ($1 \leq n \leq 10\,000$, $0 \leq c \leq 100\,000$ and $1 \leq s \leq n$): the number of places, the number of connections on the game board, and the starting place of the game, respectively.

- n lines with an integer p_i: the player controlling place i, using 0 for player even and 1 for player odd.
- c lines with two space-separated integers a and b ($1 \leq a, b \leq n$), indicating a connection from place a to place b.

Output

Per test case:

- one line with either 0 or 1, indicating the player that special agent Walker should choose in order to win.

Sample in- and output

Input	Output
3 2 1 1 0 1 1 2 6 6 1 1 0 1 1 1 0 1 2 2 3 2 5 3 4 4 6 5 6 6 7 1 1 0 0 1 1 0 1 2 1 3 1 5 2 3 3 4 3 5 5 6	1 0 0

E Encryption

Alice thinks it is very inconvenient to have to keep one of her keys in a public–private key pair secret. Therefore she invented a public–public key encryption scheme called the Really Secure Algorithm (RSA). The algorithm works as follows:

A *word* is a sequence of between one and ten capital letters (A–Z). A *sentence* is a sequence of words, separated by spaces. The *first public key* is a sentence in which each word is used at most once. The *second public key* is a sentence formed by applying a permutation σ to the words in the first public key. The *plaintext* (the unencrypted message) is a sentence that has exactly as many words as the public keys. (Unlike for the public keys, these words are not necessarily unique.) The *ciphertext* (the encrypted message) is the sentence formed by applying the permutation σ to the plaintext.

Given the two public keys and the ciphertext, recover the plaintext.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with an integer n ($1 \leq n \leq 1\,000$): the number of words in each sentence.
- one line with a sentence: the first public key.
- one line with a sentence: the second public key.
- one line with a sentence: the ciphertext.

All words consist of at least 1 and at most 10 uppercase letters.

Output

Per test case:

- one line with a sentence: the plaintext.

Sample in- and output

Input	Output
2 4 A B C D D A B C C B A P 3 SECURITY THROUGH OBSCURITY OBSCURITY THROUGH SECURITY TOMORROW ATTACK WE	B A P C WE ATTACK TOMORROW

Almost blank page

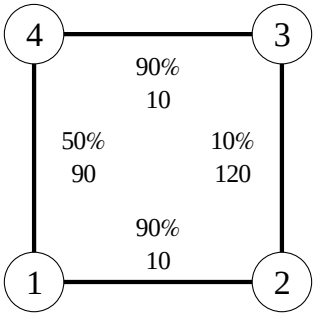
F Fare Dodging

Virgil is a loyal employee at a mysterious national security agency. He has a long commute to work by train every day, during which his mind is free to ponder his situation and the facts of life. Always the mathematician, he notices that, although he pays the fares for his journeys without fail, the conductors rarely check his tickets. In fact, through his years of experience he has a pretty good idea of what the chances are of being checked per section between each pair of cities.

Maybe he could just dodge the fares? Of course, if he gets caught in the train without a ticket he has to pay a fine that will be much higher than the cost of a ticket would have been. But still... Perhaps it's cheaper to not pay for (part of) the journey and pay the fines if and when he does get caught.

A ticket from city A to city B costs a certain start up cost s, plus a small amount p per kilometer on a shortest route between city A and B (the ticket is only valid on a shortest route). If you get caught without a ticket, you have to pay a fine consisting of a constant y plus the small amount p per kilometer on the section of railway from the last city the train visited to the next city. The regulations dictate that you have to get off at the next stop if you get caught fare dodging, but being an employee of such a mysterious agency, Virgil is a master of disguise: the conductors will never recognise him and he can continue his journey as if he was never busted.

Help Virgil write a computer program that – based on his experience – calculates the route and which tickets to buy so that his expected daily costs[1] are as small as possible.



A visual representation of the third sample. The best route from 1 to 4 is to buy a ticket for section 1 to 2 for a cost of 20 ($10 + 1 \times 10$), then dodge the fares for section 2 to 3 for an expected cost of 22 ($0.1 \times (100 + 1 \times 120)$), and finally buy a ticket for section 3 to 4 for a cost of 20 ($10 + 1 \times 10$). This leads to a total expected cost of 62 ($20 + 22 + 20$) which is the best possible for this rail network.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with seven space-separated integers:
  - n ($2 \leq n \leq 200$): the number of cities in the rail network;

[1] The expected daily costs can be calculated as $E[C] = \sum_k P_k C_k$, where P_k is the probability that the costs are C_k, and the sum is over all possible costs. Expectation values are additive, i.e. $E[X + Y] = E[X] + E[Y]$.

  - m ($1 \leq m \leq n(n-1)/2$): the number of direct connections in the rail network;
  - $start$ ($1 \leq start \leq n$): the index of the city that Virgil lives in;
  - end ($1 \leq end \leq n$, $start \neq end$): the index of the city that Virgil works at;
  - s ($1 \leq s \leq 1\,000$): the start up cost of a ticket;
  - p ($1 \leq p \leq 1\,000$): the cost per kilometer of a ticket or fine;
  - y ($s < y \leq 1\,000$): the constant part of a fine.
- m lines with four space-separated integers, describing the bidirectional sections between cities, with on line i:
  - a_i ($1 \leq a_i \leq n$): the index of one city on the end of section i;
  - b_i ($a_i < b_i \leq n$): the index of the other city of section i;
  - c_i ($0 \leq c_i \leq 100$): the probability that a conductor will check your ticket on section i, expressed as a percentage;
  - d_i ($1 \leq d_i \leq 1\,000$): the distance traveled along section i between city a_i and b_i in kilometers;

  For any two pairs (a_i, b_i) and (a_j, b_j) with $i \neq j$, we have that $(a_i, b_i) \neq (a_j, b_j)$.

Output

Per test case:

- the lowest possible expected costs for a single trip between city $start$ and end, rounded to exactly two decimals.

The test cases are such that an absolute error of at most 10^{-6} in the final answer does not influence the result of the rounding.

Sample in- and output

Input	Output
3 2 1 1 2 10 1 100 1 2 20 50 2 1 1 2 10 1 100 1 2 60 50 4 4 1 4 10 1 100 1 4 50 90 1 2 90 10 2 3 10 120 3 4 90 10	30.00 60.00 62.00

G Genuine Messages

To communicate with HQ, spies send electronic messages over the Information Superhighway using a protocol called SMTP (Secret Message Translation Protocol). To ensure that these messages are genuine and have not, for example, been sent by an evil adversary, every message is modified in such a way that it looks like there was noise on the communication line, or the sender was very nervous while typing the message. However, the mutation algorithm is carefully crafted such that an imposter is very unlikely to replicate this particular effect, and it is also easy for field agents to intentionally insert a "mistake" if they are forced at gunpoint to write a message.

In a correctly mutated message every third appearance of each letter is duplicated. For example, "`HELLOTHEREEWELLLBEFINEE`" is the correct mutation if the agent wanted to send "`HELLOTHEREWELLBEFINE`". For the past few decades these messages have been checked by highly trained monkeys. Since the number of messages arriving at the HQ has greatly increased recently, they have tasked you with writing an automated program that can alert HQ when a message is definitely fake and not sent by our agent.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with a string M ($1 \leq length(M) \leq 100\,000$), consisting of uppercase letters only: the incoming message to check.

Output

Per test case:

- one line with either "`OK`" or "`FAKE`", indicating whether or not the message M can be the result of a correctly applied mutation to some (unspecified) original message.

Sample in- and output

Input	Output
3 BAPC AABA ABCABCBBAAACC	OK FAKE OK

Almost blank page

H How Much?

This financial crisis! Even MI6's technology department does not escape the consequences. With all those constraints on the budget, how can R be expected to supply our spies with cars equipped with all the technological gadgets and weaponry they need to save the world? As if it is not hard enough to put all those tracking systems, oil sprayers, guns, bullets and missiles in one car, now R has to worry about the costs as well.

The management insists, though: before R is allowed to order a car and all the parts he needs, he will first need to determine the exact amount of money that he needs. Management will then decide whether he gets to realize his vision or not.

This sort of bureaucratic nonsense is not something that R wants to waste any time on. He therefore wants you to help him. Given the price of the car, a listing of the parts R wants to install, and the price per piece of each part, work out what the total cost is of the project.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with an integer s ($1 \leq s \leq 100\,000$): the stock price of the car.
- one line with an integer n ($0 \leq n \leq 1\,000$): the number of different parts to be installed.
- n lines with two space-separated integers q_i and p_i ($1 \leq q_i \leq 100$ and $1 \leq p_i \leq 10\,000$): for each different part i, the number of pieces q_i required of such part, and the price p_i for a single piece.

Output

Per test case:

- one line with an integer: the total amount of money needed to buy the car and all the parts.

Sample in- and output

Input	Output
2 10000 2 1 2000 3 400 50000 0	13200 50000

Almost blank page

I Identification Codes

MI6 uses a *Spy Identification Code* (SIC) to identify their spies. For example, J. B.[2] has a SIC of 7. The SICs have been assigned to the spies in such a way that MI6 can easily refer to any group of spies by using a status code that is the product of all SICs of the spies in the group. More precisely, the SICs are chosen in such a way that each status code ≥ 2 refers to a unique group of spies, and for each group of spies there is a unique status code referring to it.

Write a program that, given a status code, returns the SICs of the spies that belong to the group corresponding to that status code.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with an integer c ($2 \leq c \leq 10^9$): the status code.

Output

Per test case:

- one line with a space-separated list of SICs for the status code, in increasing order.

Sample in- and output

Input	Output
5 7 12 64 72 1337	7 3 4 4 16 2 4 9 7 191

[2]For security reasons his full name is kept secret, but rumour has it that he is one of the jury members.

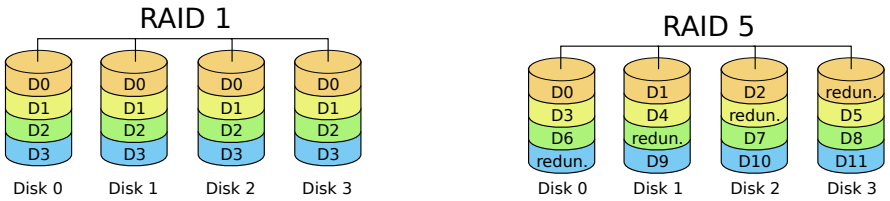
Almost blank page

J Just Enough Space

After a PR mishap with a former employee, the NSA might need to increase storage in one of their datacenters: the Russian and Spanish translators have a backlog, and the captured phone conversations need to be stored in the meantime. Up to 1 exabyte of data needs to be stored. Unfortunately, there is currently no extra storage available at all.

Due to budget limitations, it is not possible to immediately buy new disks, and the system administrator (you) wants to solve this by reducing the data redundancy. For performance and reliability, all data is currently on large RAID-1 sets of four disks in each server. More data can be stored by converting some of these sets to the slower RAID-5 technique.

Specifically, there are currently n RAID-1 sets. Each set i is built using disks of size S_i, and this set can hold S_i GB of data. If you convert one set to RAID-5, it can hold three times as much data: $3 \cdot S_i$ GB. You want to convert as few GBs of storage as possible.



Disks with size $S = 4$, capacity respectively 4 GB ($D0 \ldots D3$) and $3 \cdot 4 = 12$ GB ($D0 \ldots D11$).

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers n and e ($1 \leq n \leq 100$ and $0 \leq e \leq 10^9$): the number of RAID-1 sets, and the amount of extra space in GB required, respectively.
- one line with n space-separated integers $S_1 \ldots S_n$ ($1 \leq S_i \leq 2\,000$): the sizes of all raid sets in GB.

Output

Per test case:

- one line with an integer: the number of GB you need to convert, or the string "`FULL`" if not enough diskspace can be freed.

Sample in- and output

Input	Output
3 2 500 500 500 4 2400 400 600 700 1000 2 1000 10 10	500 1300 FULL

- In the first example, it is enough to convert one RAID-set: the new capacity is then $1500 + 500 = 2000$ GB.
- In the second example, converting the 600 GB and 700 GB disks will increase the storage from $400 + 600 + 700 + 1000 = 2700$ GB to $400 + 1800 + 2100 + 1000 = 5300$ GB, enough to store the extra 2400 GB of data. All other conversions are less efficient.
- In the third example there is no way to come up with the required amount of free disk space.

K Keys

John is on a mission to steal some documents of importance from a one-story building. He has managed to get hold of a detailed floor plan of the building, indicating the locations of all the documents. There are doors in the building that require a key to be opened. John has some keys in his possession, and there are some keys in the building itself. Can you help him figure out how many documents he can steal?

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers h and w ($2 \leq h, w \leq 100$): the height and width of the map.
- h lines with w characters describing the building:
  - `'.'` is an empty space.
  - `'*'` is an impenetrable wall.
  - `'$'` is a document that John would like to steal.
  - an uppercase letter is a door.
  - a lowercase letter is a key that opens all doors corresponding to its uppercase equivalent.
- one line with a string consisting of distinct lowercase letters: the keys that John has in his possession. If he has no keys, the line contains "`0`" instead.

John can freely move around the outside of the building. For any given door, the number of available keys that open it can be zero, one, or more than one. For any given key, the number of doors that it opens can be zero, one or more than one. Keys can be reused.

Output

Per test case:

- one line with an integer: the total number of documents that John can steal.

Sample in- and output

Input	Output
3 5 17 ***************** ....**$* *B*A*P*C**X*Y*.X. *y*x*a*p**$*$**$* ***************** cz 5 11 *.********* *...*...*x* *X*.*.*.*.* *$*...*...* *********** 0 7 7 *ABCDE* X.....F W.$$$.G V.$$$.H U.$$$.J T.....K *SQPML* irony	3 1 0