Problem A: Game

(blue balloon)

Yes, this game comes again and again. Nim is a mathematical game of strategy in which two players take turns removing objects from distinct heaps. On each turn, a player must remove at least one object, and may remove any number of objects provided they all come from the same heap. The winner of the game is the player who removes the last object.

In this problem we will play with only 3 heaps, and the initial state also will be special, it will always be in the following format (N, 2 x N, 3 x N) where N is a positive integer. For example if N is 3, the 3 heaps will initially start with 3, 6 and 9 objects.

A winning state is a state of the heaps where there's always a strategy for the player who is about to play, to win the game regardless what the other player does.

In this problem you are given two integers L and R, and your task is to find how many different values for N ($L \leq N \leq R$) such that if we use N to get the initial state as described above, it will be a winning state for the first player.

Standard Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T ($1 \leq T \leq 10^5$) representing the number of test cases. Followed by T test cases.

Each test case will be just one line containing 2 integers separated by a space, L and R ($1 \leq L \leq R \leq 2^{61}$), which are the range as described above.

Standard Output

For each test case, print a single line with the number of different values of N which satisfy the condition described above.

Note:

In the first test case, the only value for N which gives a winning state is 3, the state will be (3, 6, 9).

Sample Input	Sample Output
2 1 5 10 1000	1 854

Problem B: World cup

(red balloon)

In the world cup the teams are divided into groups, with 4 teams in each group, but in this problem we will generalize it and say the group will consist of N teams (numbered from 1 to N), and we will simulate the matches of just one group. The following are the rules of the groups stage (it's a bit different than the actual rules):

1. The group plays N - *1* rounds, with N /2 matches in each round N is guaranteed to be an even number in this problem).

2. Each team plays in one match in each round, and each pair of different teams will play against each other exactly once during the $N-1$ rounds (each team plays against the other $N-1$ teams).

3. In each match, the winning team gets 3 points and the losing team gets 0 points, or both will get 1 point if the match ends in draw.

4. After the N - *1* rounds, they calculate how many points each team got in total, and the top 2 teams with the highest number of points will be qualified to the next stage (note that it's always top 2 teams even if N is greater than 4).

5. All teams with the same number of points like one of the top 2 teams will be qualified as well (we don't consider the scored goals in this problem). Which means it's possible that the whole group might be qualified to the next stage (if they all get the same number of points in total).

You are given the results of the first N - *2* rounds of one group, and your task is to calculate for each team if they still have a chance to be qualified to the next round or if they are disqualified for sure no matter what.

Standard Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer T ($1 \le T \le 100$) representing the number of test cases. Followed by T test cases.

Each test case starts with a line containing an integer N *($4 \le N \le 1{,}000$)* representing the number of teams followed by (N - 2) * (N /2) lines, which are the results of the first N - *2* rounds, each line is the result of one match.

Each match result line will start with 2 integers A and B separated by a space *($1 \le A < B \le N$)*, then another space and a character which will be either '`W`' (means team A won), '`L`' (means team B won) or '`D`' (means the match ended in draw).

Standard Output

For each test case print a single line containing a string of *N* characters, the first character is for team 1, the second character is for team 2 and so on. The character should be '1' if the corresponding teams might still be qualified to the next round, or '0' if that team is disqualified for sure no matter what.

Note:

In the first test case, teams 1 and 3 got 6 points so far, and teams 2 and 4 got 0 points. No matter what happens in the last round, teams 2 and 4 can't be in the top 2 teams.

In the second test case, all matches ended in draw, so all teams still have a chance to be qualified.

Sample Input	Sample Output
2 4 1 2 W 3 4 W 1 4 W 2 3 L 6 1 2 D 1 3 D 1 4 D 1 5 D 2 3 D 2 4 D 2 6 D 3 5 D 3 6 D 4 5 D 4 6 D 5 6 D	1010 111111

Problem C: LCP

(yellow balloon)

In this problem you are given an array of strings, these strings are given unique indexes from 1 to N (in the same order as in the input). Then you are given Q queries, each query consists of 2 integers L and R, to answer the query you need to find a pair of strings with different indexes in the range from L to R (inclusive), where the length of the longest common prefix for these 2 strings is the maximum among all other possible pairs.

Standard Input

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

Each test case starts with a line containing an integer N *($2 \leq N \leq 10^5$)* representing the number of strings followed by a line containing N non-empty strings of lower case English letters separated by a single space, representing the list of strings. The sum of lengths of the strings in each test case is not greater than 200,000.

Followed by a line containing an integer Q *($1 \leq Q \leq 10^5$)* representing the number of queries followed by Q lines, each line will contain 2 integers separated by a space, 'L R', which represent a query as described above *($1 \leq L < R \leq N$)*.

Standard Output

For each query print a single line containing an integer which is the maximum length of a longest common prefix as described above.

Note:

A prefix of string S is the first (from the left) 0 or more characters from S, and a common prefix between 2 strings is a string which is a prefix in both of them.

Sample Input	Sample Output
1 4 aab abc aac xba 3 2 3 1 3 3 4	1 2 0

Problem D: Adam (Orange balloon)

Adam has just started learning how to walk (with some help from his brother Omar), and he falls down a lot. In order to balance himself, he raises his hands up in the air (that's a true story), and once he puts his hands down, he falls.

You are given a string, each character represents a step he walks, if that character is '`U`' that means his hands are up in this step, if this character is '`D`' that means his hands are down and he fell down in this step. Your task is to count how many steps he will walk before falling down for the first time.

Standard Input

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

Each test case will consist of a single line, containing a non-empty string of at most 100 characters, and each character is either '`U`' or '`D`'. The characters from left to right represent Adam's steps in the order he walks them.

Standard Output

For each test case print a single line containing the number of steps that Adam will walk before falling down, or the length of the string if he won't fall down.

Note:

In the first test case, he falls down after 3 steps.
In the second test case, he falls down before making any steps.
In the third test case, he doesn't fall down at all.

Sample Input	Sample Output
3 UUUDU DDD UU	3 0 2

Problem E : Stock

(purple balloon)

You have just started investing in a stock, and an insider told you that there's a list of N prices (in dollars) which will be assigned to that stock in the next N days (in some order which no one knows yet). Each price in that list will be assigned to exactly one of the next N days, and the price within the same day won't change. Your insider friend accepted to do any assignments you want, so you want to assign the prices to the days to maximize your profit.

Let's say you have already assigned the N prices to the N days. Before day 1, you already own 1 unit of that stock and you have no money at all. In each of the following N days, you can either sell (might be partial units) some of the stocks you own (and you get money in dollars), or you can buy (might be partial units) some of the available stocks (and you pay in dollars), using the price assigned to that day. You can also do nothing in any given day. Assume that there's an infinite supply of stocks in each day.

Your profit at the end is the amount of money you have, so you need to sell everything by the end of the N days.

Given the list of prices, your task is to arrange them somehow as described above to get the maximum profit.

Standard Input

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

Each test case starts with a line containing an integer N $(1 \leq N \leq 10^5)$ representing the number of days followed by a line containing N positive integers separated by a single space, representing the list of prices you will assign, each price will be at most 10^6.

Standard Output

For each test case print a single line containing a single decimal number rounded to exactly 6 decimal places, which is the maximum profit you can get. It's guaranteed that the result will not be more than 10^{18}.

Note:

In the first test case, you assign the prices in this order '`2 1 3`', in the first day you sell the unit you have for 2 dollars, in the second day you buy 2 units for 1 dollar each, and in the third day you sell the 2 units for 3 dollars each and you end up with 6 dollars profit.

In the second test case, you have just 1 day and you must sell the unit you have in that day.

Sample Input	Sample Output
2 3 1 2 3 1 100	6.000000 100.000000

Problem F : Files

(green balloon)

Many of us have this problem on their computer, where we make several copies of the exact same files, which uses more memory. So you finally decided to write a program to remove all duplicate files.

Here's how your computer works. Whenever you create a new file, it gets a unique ID (when you make a copy of an existing file, the new copy gets a new ID). The IDs are relative to the time, so older files get smaller IDs (but the IDs are not necessary sequential). Also each file has a name, but multiple files can have the same name, and when 2 or more files get the same name, this means they are all exactly the same and they are just identical copies of the one of them with the smallest ID.

You are given the list of all files with their names and IDs, your task is to delete all duplicates and just keep the oldest copy of each file.

Standard Input

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

Each test case starts with a line containing an integer N $(1 \leq N \leq 10^5)$ representing the number of files followed by N lines, each line will contain the file name followed by a space then the file ID. The file name is a non-empty string of at most 10 lower case English letters, and the ID is a positive integer which is at most 105.

All IDs will be distinct within each test case.

Standard Output

For each test case print a single line containing the IDs of the files which won't get deleted, the IDs should be sorted in increasing order and separated by a single space.

Sample Input	Sample Output
2 2 aaa 6 aa 5 3 file 3 file 2 file 1	5 6 1