Problem A. Alien invaders

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7.8 seconds
Memory limit:	512 mebibytes

The aliens from outer space have (finally!) invaded Earth. Defend yourself, or be disintegrated! Or assimilated. Or eaten. We are not yet sure.

The aliens follow a known attack pattern. There are n attackers, the i-th one appears at time a_i, at distance d_i from you. He must be destroyed no later than at time b_i, or else he will fire his weapon, which will definitely end the fight.

Your weapon is an area-blaster, which can be set to any given power. If fired with power R, it momentarily destroys all aliens at distance R or smaller. It also consumes R fuel cells.

Determine the minimal cost (measured in fuel cells) of destroying all the aliens, without being killed.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case starts with a line containing the number of aliens n ($1 \le n \le 300$). Of the next n lines, the i-th one contains three integers a_i, b_i, d_i, ($1 \le a_i < b_i \le 10\,000$; $1 \le d_i \le 10\,000$). The i-th alien appears at time a_i, is idle until b_i, and his distance from you is d_i.

Output

For each test case, output one line containing the minimum number of cells needed to destroy all the aliens.

Example

standard input	standard output
1 3 1 4 4 4 7 5 3 4 7	7

Problem D. Dictionary

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3.1 seconds
Memory limit:	512 mebibytes

According to a popular belief, computer programmers drink a lot of coffee and know only a few words. The vocabulary of a typical programmer consists of just three words. Besides, he rarely knows how to spell them. To help programmers with their spelling mistakes, we published a book titled «The Dictionary of the Three Words Every Typical Programmer Should Know».

You got a copy of the book but, soon after that, you spilled your coffee over it. Now, you cannot read some of the characters. Fortunately, the three words were, as usually in dictionaries, distinct and printed in lexicographical order.

Before you attempt to use that fact to recover the missing characters, you want to know in how many different ways you can do it. Since you expect this number might be large, you want to know it modulo $10^9 + 9$.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case consists of three lines, each containing a single nonempty word – in the order they appear in the dictionary. Words consist of small letters of the English alphabet and quotation marks, the latter denoting missing characters. Each word is at most 1 000 000 characters long.

Output

For each test case, output one line containing the number of different ways you can substitute each question mark with one of the 26 letters from '`a`' to '`z`' in such a way that the three words are distinct and in lexicographical order. The number should be printed modulo $10^9 + 9$.

Example

standard input	standard output
3 ?heoret?cal c?mputer ?cience jagiellonian ?niversity kra?ow ? b c	42562 52 1

Problem E. Express As The Sum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Given an integer N, express it as the sum of at least two consecutive positive integers. For example:

$10 = 1 + 2 + 3 + 4$

$24 = 7 + 8 + 9$

If there are multiple solutions, output the one with the smallest possible number of summands.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case consists of one line containing an integer N ($1 \le N \le 10^9$).

Output

For each test case, output a single line containing the equation in the format:

`N = a + (a+1) + ...+ b`

as in the example. If there is no solution, output a single word "`IMPOSSIBLE`" instead.

Example

standard input	standard output
3 8 10 24	IMPOSSIBLE 10 = 1 + 2 + 3 + 4 24 = 7 + 8 + 9

Problem F. Factory

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

A very important and complicated machine on the factory consists of n wheels, numbered $1, 2, \ldots, n$. They are actually cogwheels, but the cogs are so small that we can model them as circles on the plane. Every wheel can spin around its center.

Two wheels cannot overlap (they do not have common interior points), but they can touch. If two wheels touch each other and one of them rotates, the other one spins as well, as their micro-cogs are locked together.

A force is put to wheel 1 (and to no other wheel), making it rotate at the rate of exactly one turn per minute, clockwise. Compute the rates of other wheels' movement. You may assume that the machine is not jammed (the movement is physically possible).

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case consists of one line containing the number of wheels n ($1 \le n \le 1000$) . Each of the following lines contain three integers x, y and r ($-10\,000 \le x, y \le 10\,000$; $1 \le r \le 10\,000$) where (x, y) denote the Cartesian coordinates of the wheel's center and r is its radius.

Output

For each test case, output n lines, each describing the movement of one wheel, in the same order as in the input. For every wheel, output either "`p/q clockwise`" or "`p/q counterclockwise`", where the irreducible fraction p/q is the number of wheel turns per minute. If q is 1, output just p as an integer. If a wheel is standing still, output "`not moving`".

Example

standard input	standard output
1 5 0 0 6 6 8 4 -9 0 3 6 16 4 0 -11 4	1 clockwise 3/2 counterclockwise 2 counterclockwise 3/2 clockwise not moving

Problem H. How to Deal With Imp

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

You arrive in Ye Olde Magic Shoppe with some hard-earned gold to purchase wondrous and unique magic items. There are n such items in the shop, each of them locked in a special magic box. The i-th box costs c_i gold pieces to buy, and contains an item worth v_i gold pieces. The costs and item values are known to you, as you have previously read, mastered, and memorized Ye Olde Magic Catalogue.

A mortal, such as you, can safely carry only one magic item. You therefore aim to get the most precious one. And obtain it you would, if not for a malicious, magical creature, known as The Imp.

The Imp can cast a mischievous spell, which transforms the content of any magic box into worthless dust. Of course, he will use the spell just after you buy a box, to make you pay for the item and not get it. You are thus forced to buy another box, and then the next one...

The Imp has enough magic to cast the spell at most k times. He can, of course, refrain from using it, allowing you to keep an item. You can walk away at any time, empty-handed (though it would surely be a disgrace). However, if you get an item, you must keep it and leave the shop. You aim to maximize your gain (the value of the acquired item minus all the expenses paid previously), while The Imp wants to minimize it. If both you and the creature use the optimal strategy, how much gold will you earn?

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case starts with a line containing the number of items n ($1 \le n \le 150\,000$) and the the maximum number of The Imp's spells k ($0 \le k \le 9$). The next n lines contain the items' values and costs, the i-th line containing the numbers v_i and c_i, in that order ($0 \le v_i, c_i \le 10^6$).

Output

For each test case, output one line containing your gain.

Example

standard input	standard output
1 3 1 10 5 8 1 20 12	7

Problem J. Joining The Bricks Together

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15.6 seconds
Memory limit:	512 mebibytes

You are given a sequence of white (W) and black (B) bricks. The goal is to partition it into some number of non-empty, contiguous blocks, each one having the same ratio of white and black bricks.

Of course one can always «partition» the sequence into one single block (which is not very interesting). We want, however, to have as many blocks as possible. Consider for example the following sequences and its partitions:

`BWWWBB = BW + WWBB` (ratio 1:1),

`WWWBBBWWWWWWWWWB = WWWB + BBWWWWWW + WWWB` (ratio 3:1).

Note that both of these partitions are optimal with respect to the number of blocks.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case starts with one line containing an integer n ($1 \le n \le 10^5$) which is the length of the description of a sequence. Each of the following n lines consists of an integer k ($1 \le k \le 10^9$) and one of the characters `W` or `B`, meaning that k bricks of the given color follow next in the sequence. It is guaranteed that the total length of the brick sequence does not exceed 10^9.

Output

For each test case, output a single line containing the largest possible number of blocks.

Example

standard input	standard output
3 3 1 B 3 W 2 B 4 3 W 3 B 9 W 1 B 2 2 W 3 W	2 3 5

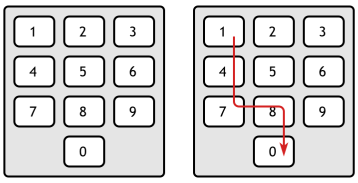
Problem K. Keyboard Troubles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Good morning! This is your 5am wake-up call! A partly cloudy day is expected with light rain coming afternoon...

You have just woken up. You desperately need coffee... and... more coffee... and some cereal. And your clothes. And coffee.

To prepare warm cereal, you put some milk into a microwave, trying to heat it for k seconds. You must enter k on the microwave keyboard:



As you still haven't had your coffee, your hand (along with eyes and brain) keeps falling down. You are only able to enter a number if your hand would only move downwards and/or to the right. You cannot go back left, nor move your hand up, though you can press the same key again. And again... and again...

For example, you can enter the number 180 or 49, but not 98 or 132. Enter a number that is as close to k as possible. If there are two solutions, enter any one of them. You are too sleepy to actually care. And you need coffee.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case consists of one line containing an integer k ($1 \le k \le 200$).

Output

For each test case, output a number that is closest to k which can be entered on the keyboard.

Examples

standard input	standard output
3 180 83 132	180 80 133

Problem L. Factory

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

N machines are installed on the factory. Each machine produces identical details. For each machine is given time in the minutes to produce one detail. Machines are working independently from each other. Find out mininal time in minutes, needed for production of K details; you may use one machine as well as several simultaneously working ones.

Input

First line of the input contains one integer T — number of the test cases ($1 \le T \le 700$). First line of the test case contains two integers: N — number of machines ($1 \le N \le 10^5$) and K — number of details to produce ($1 \le K \le 10^9$). Second line contains N positive integers t_i — time in minutes per one detail for i'th machine. All t_i does not exceed 10^9, sum of all N's in the input file does not exceed $2 \cdot 10^9$.

Output

For each test case print in the separate line one integer — minimal time in minutes, needed to produce K details.

Example

standard input	standard output
1 2 1 2 2	2

Problem M. C--

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

C-- language was designed specifically for students who failed standard course of Compiler Programming.

C-- program is a list of statements. Each statement is one of:

-
- `-- start of the block;`
- — end of the block;
- `int` *var* — variable declaration;
- var_1=var_2 — assign var_2 to var_1;
- *var*=*constant* — assign *constant* to *var*;
- `print` *var* — print value of *var*.

Variable names are single small Latin letters. Constants are integers in range from 0 to 10^9. There are no spaces anywhere except after "`int`" and "`print`" keywords.

Each variable is declared at most once in the block, but variables with same name may be declared in different blocks. It that case, name refers to the variable declared in the nearest enclosing block. Note that variables may be defined in the middle of the block.

The program is guaranteed to be correct — all blocks are properly balanced, variables are referenced only after declaration, variables are always initialized before reading, etc. Program contains at least one "`print`" statement.

Your program must execute a given C-- program.

Input

First line of input file contains integer N ($3 \le N \le 1000$). Following N lines contain one C-- statement each.

Output

Output file must contain a sequence of integers — one integer for each "`print`" statement.

Example

standard input	standard output

10
int a
int x
{
a=50
int a
a=60
x=a
print x
}
print a
	60
	50

Problem N. Allo

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Given a string, consisting of digits from 0 to 9. Find out all digits, which are not used in this string.

Input

First line of the input file contains one integer T — number of the test cases ($1 \le T \le 150$). Each of next T lines contains one test case — non-empty string, consisting of digits from 0 to 9. Length of the given string does not exceed 30.

Output

For each test case print in accending order all digits, which did not used in this string. If all ten digits are used, print "`allo`" instead.

Examples

standard input	standard output
2 2468 0123456789	013579 allo

Problem O. Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Alice and Bob are playing in the next game. Initially, number 1 is given. On his turn each player must multiply current number by one of integers between 2 and 9, inclusively. Goal is to obtain number not less than given integer N. Player, who obtained such a number first, is declared as winner. Alice always starts first.

Find out, who will win if Alice and Bob will play optimally.

Input

First line of the input contains one integer T — number of the test cases ($1 \le T \le 2500$). Each of next T lines contain one integer N ($2 \le N \le 10^9$).

Output

For each test case print in separate line 1, if Alice will win the game, and 2 otherwise.

Example

standard input	standard output
4 9 10 1149729 999999999	1 2 2 1