Problem Set Overview

This contest has **13** problems, each worth **100** points each. Most problems consist of two or more subtasks. Note that problems are not guaranteed to be sorted in ascending order of difficulty.

ID	Problem Name	Std. Time Limit
A	!(Hello World)	1 s
B	Sum Random Problem	1 s
C	Nonclassic 3n+1	1 s
D	DAG	1 s
E	Count LIS	1 s
F	The Creeper Crisis	1 s
G	Splitaire	1 s
H1	The Lazy Problem (easy version)	1.5 s
H2	The Lazy Problem (hard version)	1.5 s
I	Square Distance	1.5 s
J	K4C	1.5 s
K1	Math 101 (easy version)	1 s
K2	Math 101 (hard version)	1 s

!(Hello World)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

NOT Hello World.

Input

No input.

Output

NOT Hello World.

Scoring

This problem is worth 100 points. There are no subtasks.

Sum Random Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

To sum up, this problem is about.

Input

Each test contains multiple test cases. The first line of input contains a single integer T ($1 \le T \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains an integer N ($1 \le N \le 10^5$) — describing how many numbers to add.

The next line contains N integers $a_1, a_2, \ldots a_N$ ($|a_i| \le 10^9$) — the numbers.

It is guaranteed that the sum of N over all test cases does not exceed 10^5.

Output

For each test case, output the sum of the given numbers on a separate line.

Scoring

This problem is composed of 3 subtasks:

Subtask	Add'l Constraints	Points
1	Sample testcases	4
2	$\|x\| \le 10^4$	32
3	None	64

Example

standard input	standard output
3 2 1 1 10 -1 -2 -4 -8 -16 -32 -64 -128 -256 -512 5 1000000000 999999999 999999998 999999997 999999996	2 -1023 4999999990

Note

In the first test case, it can be shown that $1 + 1 = 2$.

In the second test case, we can use the geometric series formula to calculate the sum:

$$S = a_1 \frac{1 - r^n}{1 - r} = (-1)\frac{1 - 2^{10}}{1 - 2} = -1\,023$$

In the third test case, we can use the arithmetic series formula to calculate the sum:

$$S = \frac{n(a_1 + a_n)}{2} = \frac{5(1\,000\,000\,000 + 999\,999\,996)}{2} = 4\,999\,999\,990$$

Nonclassic 3n+1

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

The classic $3n + 1$ problem (aka Collatz conjecture) involves repeatedly applying a simple rule to an integer n:

- If n is even, divide it by 2.
- If n is odd, multiply it by 3 and add 1.

Novice programmer script_kidd was playing around with this problem the other day. He implemented a JajaScript program to see how a value n would change after k iterations. (please refer to web version for pseudocode)

Surprisingly, when script_kidd ran the program with inputs $n = 10$ and $k = 2$, he got 1111 instead of the expected 16 (see **Note** section)! Later, script_kidd realized that he accidentally passed n as a string argument instead of an integer. The code still compiled — because in JajaScript:

- When adding an integer a to a string s, it is appended to the end of s.
- When multiplying a string s by an integer a, it becomes $\underbrace{s + s + \cdots + s + s}_{a \text{ times}}$.
- When dividing a string s by 2, the **latter** half of the string is removed. If the length of the string is odd, the middle character is not removed.
- A string s is even if its numerical equivalent is even.

PR0C0D3R is intrigued by script_kidd's program but doesn't have access to it. So help PR0C0D3R out!

Input

Each test contains multiple test cases. The first line of input contains a single integer T ($1 \le T \le 20$) — the number of test cases. The description of the test cases follows.

The first and only line contains two integers n and k ($1 \le n \le 10^5, 1 \le k \le 9$) — the input to script_kidd's program.

Output

For each testcase, print the output of script_kidd's program on a separate line.

Scoring

This problem is composed of 3 subtasks:

Subtask	Add'l Constraints	Points
1	$k = 1$	40
2	$k \le 6$	40
3	None	20

Example

standard input	standard output
3 10 2 22 3 999 1	1111 2 9999999991

Note

For the first testcase, initially $n = 10$. The program goes through $k = 2$ iterations:

1. $n = 10$ is even, so "divide by 2". n becomes 1.
2. $n = 1$ is odd, so "multiply by 3" and "add 1". n becomes 1 111.

For the second testcase, $22 \to 2 \to 2 \to 2$.

For the third testcase, $999 \to 9\,999\,999\,991$.

DAG

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Topological sort might help.

— Someone, probably

Kyle_Ho is an expert in sequences. One day, he came up with the following problem and challenges you to solve it.

Given a sequence of nonnegative integers, determine if it is arithmetic-geometric. A sequence s is arithmetic-geometric if **any** consecutive ordered triplet in s forms an arithmetic progression[†] **or** geometric progression[‡].

[†] Can be written as $(a, a+d, a+2d)$

[‡] Can be written as (a, ar, ar^2) where $a \neq 0$ and $r \neq 0$

Can you cope with Kyle_Ho's challenge?

Input

Each test contains multiple test cases. The first line of input contains a single integer T ($1 \leq T \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains one integer N ($3 \leq N \leq 10^5$) — the length of the sequence.

The second line contains N integers $a_1, a_2, \ldots, a_N$ ($0 \leq a_i \leq 10^9$) — the sequence.

It is guaranteed that the sum of N over all test cases does not exceed 10^5.

Output

For each test case, output "`YES`" if the sequence is arithmetic-geometric and "`NO`" otherwise.

You can output "`YES`" and "`NO`" in any case (for example, "`yEs`", "`yes`" and "`Yes`" will be recognized as positive responses).

Scoring

This problem is composed of 3 subtasks:

Subtask	Add'l Constraints	Points
1	$N = 3$	33
2	$a_i \leq 10^4$	33
3	None	34

Example

standard input	standard output
3 4 40 30 20 10 5 2 4 6 9 12 3 1 0 0	YES YES NO

Note

In the first test case, since $(40, 30, 20)$ and $(30, 20, 10)$ are both arithmetic sequences with $d = -10$, the answer is `YES`.

In the second test case, the answer is `YES` because:

- $(2, 4, 6)$ is arithmetic ($d = 2$)
- $(4, 6, 9)$ is geometric ($r = \frac{3}{2}$)
- $(6, 9, 12)$ is arithmetic ($d = 3$)

In the third test case, since $(1, 0, 0)$ is neither arithmetic nor geometric, the answer is `NO`.

Count LIS

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A *subsequence* of the sequence A comes from removing zero or more elements of A, without changing the order of the elements. For example, if $A = [10, 40, 20, 30, 50]$, then some valid subsequences include:

- $[10]$
- $[10, 20, 30]$
- $[40, 50]$
- $[10, 40, 20, 30, 50]$

For his birthday, script_kidd received an array p of length N, consisting of $[1, 2, \ldots, N]$. He plans to rearrange its elements to **maximize** the **number** of longest increasing subsequences in p. Before that, he wants to know what this maximum is.

You know the answer, but decide to only tell script_kidd its remainder when divided by $10^9 + 7$. So what will you tell him?

Input

Each test contains multiple test cases. The first line of input contains a single integer T ($1 \leq T \leq 100$) — the number of test cases. The description of the test cases follows.

The first and only line contains an integer N ($1 \leq N \leq 100$) — the length of p.

Output

For each test case, print the desired maximum modulo $10^9 + 7$ on a separate line.

Scoring

This problem is composed of 4 subtasks:

Subtask	Add'l Constraints	Points
1	$N \leq 4$	1
2	$N \leq 5$	33
3	$N \leq 20$	33
4	None	33

Example

standard input	standard output
5 1 2 3 4 69	1 2 3 4 143178169

Note

When $n = 1$, it is optimal to rearrange p into $[1]$.

When $n = 2$, it is optimal to rearrange p into $[2, 1]$.

When $n = 3$, it is optimal to rearrange p into $[3, 2, 1]$.

When $n = 4$, there are two optimal ways to rearrange p:

1. $[4, 3, 2, 1]$: [4], [3], [2], [1] are longest increasing subsequences.
2. $[2, 1, 4, 3]$: [2, 4], [2, 3], [1, 4], [1, 3] are longest increasing subsequences.

Both result in 4 longest increasing subsequences, which is the maximum.

When $n = 69$, note that the actual maximum is much greater than 143 178 169. This value is its remainder when divided by $10^9 + 7$.

The Creeper Crisis

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

sorry if the gif is broken >w<

On a $H \times W$ map, creepers are commanded to move in one direction at 1 block per second. They aren't exactly chill dudes though. When two creepers collide **oppositely**—kaboom![†] Even worse, there may be TNT blocks lying around that will detonate upon impact with a creeper.[‡]

Alex is on a mission to prevent chaos (and save her CPU). But as you know, fighting creepers is no easy task. Alex needs your help to determine the minimum number of creepers she needs to kill to prevent **any** explosions from happening.

[†] When creepers collide perpendicularly, nothing happens.

[‡] Blame her friend Steve for installing the CursedCreeper mod.

Input

Each test contains multiple test cases. The first line of input contains a single integer T ($1 \le T \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains two integers H and W ($1 \le H, W \le 10^6$) — the number of rows and columns in the map.

The following H lines each contain W characters — a *block* listed below. Together, they form the map.

Block Type	Character	Description
Air	`.`	A block creepers can freely pass through
TNT	`T`	A block that detonates when hit by a creeper
Creeper	`L,R,U,D`	A creeper headed left, right, up, or down

It is guaranteed that the map always contains at least one creeper. Furthermore, the sum of $H \times W$ over all test cases does not exceed 10^6.

Output

Output the minimum number of creepers Alex needs to kill to neutralize the map.

Scoring

This problem is composed of 3 subtasks:

Subtask	Additional Constraints	Points
1	There are exactly two creepers	33
2	$H, W \leq 10$	33
3	None	34

Example

standard input	standard output
3 3 9 RDLL..RRL .D..U.RD. ..LTTTTTT 5 5 ..... .LLU. .D.U. .DRR. ..... 5 5 TTTTT TLLUT TD.UT TDRRT TTTTT	3 0 8

Note

Let (r, c) represent the position at row r and column c.

In the first test case, Alex should eliminate the creepers originally at $(1, 1)$ and $(1, 9)$ to prevent them from colliding with others. She should also eliminate the creeper at $(2, 8)$, or else it will detonate the TNT block below. Therefore, the answer is 3.

Note that it's fine that the creepers at $(2, 2)$ and $(3, 3)$ collide, as they aren't going in opposite directions.

Because of the large input, it is recommended to use fast I/O for this problem to avoid unjustified TLEs. Feel free to use the attached templates in the "contest materials" section.

Splitaire

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Welcome to K4Casino! Let's play *Split*aire. Here are the rules:

1. The dealer lays out A in front of you, a deck of unique cards.
2. You can split A into several parts. Splitting cards at the i-th gap will gain you C_i dollars. Beware, some splits could cost you.
3. You can now shuffle the **parts**[†] however you like.
4. The dealer then reveals B, another deck of unique cards.
5. In the end, if your deck of cards matches B exactly, you gain a bonus[‡] of X dollars.

Plot twist: You, a mastermind, have somehow figured out B in advance. What is your net gain if you play optimally?

[†] You may not rearrange cards within a part. Only the parts themselves can be shuffled.

[‡] Could be negative. Don't ask why.

Input

Each test contains multiple test cases. The first line of input contains a single integer T ($1 \le T \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains two integers N and X ($2 \le N \le 10^5, -10^9 \le X \le 10^9$) — the number of cards in each deck and the "bonus".

The second line contains N unique integers $A_1, A_2, \cdots, A_N$ ($1 \le A_i \le N$) — the cards of A in order.

The third line contains N unique integers $B_1, B_2, \cdots, B_N$ ($1 \le B_i \le N$) — the cards of B in order.

The last line contains $N-1$ integers $C_1, C_2, \cdots, C_{N-1}$ ($-10^4 \le C_i \le 10^4$) — the gains associated with splitting the deck at each gap.

It is guaranteed that the sum of N over all test cases does not exceed 10^5.

Output

For each test case, output the optimal net gain.

Scoring

This problem is composed of 5 subtasks:

Subtask	Additional Constraints	Points
1	$X = 0$	20
2	$B = [1, 2, \ldots, N]$ and $X \ge 0$	20
3	$B = [1, 2, \ldots, N]$	20
4	The sum of N over all test cases ≤ 1000	20
5	None	20

Example

standard input	standard output
1 4 10 4 1 2 3 1 2 3 4 -5 3 -10	8

Note

Please refer to the web version for the explanation of the test cases.

The Lazy Problem (easy version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

I was too lazy to think of a statement so yea :)

— Problemsetter

This is the easy version of the problem. The two versions only differ in the constraints of the input and subtask #5.

In this problem, the length of a string s is denoted by $|s|$.

Given a string s, perform the following types of operations:

- INS $pos\,t$: An integer pos and a string t are given. Insert t at the front, middle, or end of s depending on pos $(-1, 0, 1)$. If $pos = 0$, it is guaranteed that $|s|$ is **even** immediately before the operation.
- REPL $a\,b$: Two distinct characters a and b are given. Replace every a with b in s, if any.
- REV: Reverse s.
- SORT: Sort s in the following alphabetical order: `qwertyuiopasdfghjklzxcvbnm`.

Output s after all operations.

Input

Each test contains only one test case.

The first line contains the string s ($1 \le |s| \le 2000$) — the original string.

The following Q ($1 \le Q \le 2000$) lines describe the operations. Each line starts with an operation $\text{op} \in \{\text{INS}, \text{REPL}, \text{REV}, \text{SORT}\}$, followed by corresponding parameters.

It is guaranteed that all input (excluding the operation names) consists only of lowercase English letters, and that the sum of $|t|$ does not exceed 2000.

Output

Output a single line containing s after all operations.

Scoring

This problem is composed of 6 subtasks:

Subtask	Additional Constraints	Points
1	$\text{op} = \text{INS}$ and $pos \in \{-1, 1\}$	10
2	$\text{op} = \text{INS}$	10
3	$\text{op} = \text{REPL}$	10
4	$\text{op} \in \{\text{INS}, \text{REPL}, \text{REV}\}$	30
5	$Q = 1$	20
6	None	20

Examples

standard input	standard output
thequickbrownfoxjumpsoverthelazydog SORT	qweeerrttyuuiooooopasdfghhjklzxcvbnm
kcis REV REV REV	sick
aa REPL b x INS -1 ab INS 1 ba INS 0 cbc REPL a o REPL b w REPL c u	owouwuowo

Note

Explanation for the third test case:

1. Replace all `b` with `x`. As there is no `b`, s remains `aa`.
2. Insert `ab` at the front. s becomes `abaa`.
3. Insert `ba` at the end. s becomes `abaaba`.
4. Insert `cbc` in the middle, between `aba` and `aba`. s becomes `abacbcaba`.
5. Replace all `a` with `o`. s becomes `obocbcobo`.
6. Replace all `b` with `w`. s becomes `owocwcowo`.
7. Replace all `c` with `u`. s becomes `owouwuowo`.

Therefore, the answer is `owouwuowo`.

The Lazy Problem (hard version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

I was too lazy to think of a statement so yea :)

— Problemsetter

This is the hard version of the problem. The two versions only differ in the constraints of the input and subtask #5.

In this problem, the length of a string s is denoted by $|s|$.

Given a string s, perform the following types of operations:

- INS $pos\,t$: An integer pos and a string t are given. Insert t at the front, middle, or end of s depending on pos $(-1, 0, 1)$. If $pos = 0$, it is guaranteed that $|s|$ is **even** immediately before the operation.
- REPL $a\,b$: Two distinct characters a and b are given. Replace every a with b in s, if any.
- REV: Reverse s.
- SORT: Sort s in the following alphabetical order: `qwertyuiopasdfghjklzxcvbnm`.

Output s after all operations.

Input

Each test contains only one test case.

The first line contains the string s ($1 \le |s| \le 5 \cdot 10^5$) — the original string.

The following Q ($1 \le Q \le 5 \cdot 10^5$) lines describe the operations. Each line starts with an operation $\text{op} \in \{\text{INS}, \text{REPL}, \text{REV}, \text{SORT}\}$, followed by corresponding parameters.

It is guaranteed that all input (excluding the operation names) consists only of lowercase English letters, and that the sum of $|t|$ does not exceed $5 \cdot 10^5$.

Output

Output a single line containing s after all operations.

Scoring

This problem is composed of 6 subtasks:

Subtask	Additional Constraints	Points
1	op = INS and $pos \in \{-1, 1\}$	10
2	op = INS	10
3	op = REPL	10
4	$\text{op} \in \{\text{INS}, \text{REPL}, \text{REV}\}$	30
5	$\text{op} \in \{\text{INS}, \text{REV}, \text{SORT}\}$	20
6	None	20

Examples

standard input	standard output
thequickbrownfoxjumpsoverthelazydog SORT	qweeerrttyuuiooooopasdfghhjklzxcvbnm
kcis REV REV REV	sick
aa REPL b x INS -1 ab INS 1 ba INS 0 cbc REPL a o REPL b w REPL c u	owouwuowo

Note

Explanation for the third test case:

1. Replace all `b` with `x`. As there is no `b`, s remains `aa`.
2. Insert `ab` at the front. s becomes `abaa`.
3. Insert `ba` at the end. s becomes `abaaba`.
4. Insert `cbc` in the middle, between `aba` and `aba`. s becomes `abacbcaba`.
5. Replace all `a` with `o`. s becomes `obocbcobo`.
6. Replace all `b` with `w`. s becomes `owocwcowo`.
7. Replace all `c` with `u`. s becomes `owouwuowo`.

Therefore, the answer is `owouwuowo`.

Because of the large input, it is recommended to use fast I/O for this problem to avoid unjustified TLEs. Feel free to use the attached templates in the "contest materials" section.

Square Distance

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

PureDevr has always been amazed by perfect squares, those numbers that stand out as the squares of integers. For example, $0 = 0^2$, $1 = 1^2$, $4 = 2^2$, and $5221225 = 2285^2$ are perfect squares, whereas -1, 10, and 69 are not. Here is his favorite fact: Did you know that every positive integer can be represented as the sum of four perfect squares?

One night, the following problem came to PureDevr in a dream:

Define the $f(x)$ as the minimum distance between x and a perfect square. Formally,

$$f(x) = \min_{n \in \mathbb{Z}} |x - n^2|$$

Given two integers l and r, find a **nonempty subarray** of $[l, l+1, \ldots, r]$ such that:

- Let p be the product of all numbers in the subarray.
- $m = f(p)$ is minimized. Report this m.

PureDevr has prepared multiple test cases to see if you can rise to the challenge. Can you do it?

Input

The first line contains an integer T ($1 \le T \le 10^4$) — the number of testcases. The description of the test cases follows.

The first and only line contains two integers l and r ($1 \le l \le r \le 10^6$) — the given range.

Output

For each test case, print the desired m on a separate line.

Scoring

This problem is composed of 2 subtasks:

Subtask	Add'l Constraints	Points
1	$r \le 20$	20
2	None	80

Example

standard input	standard output
3 3 5 70 80 1971 1973	0 1 35

Note

In the first test case, we should choose the *subarray* $[4, 4]$ where $m = |4 - 2^2| = 0$, which is obviously optimal.

In the second test case, we can choose the *subarray* $[80, 80]$ where $m = |80 - 9^2| = 1$. It can be shown that this value of m is optimal.

In the third test case, it is optimal to choose the *subarray* $[1971, 1973]$ where $m = |1971 \cdot 1972 \cdot 1973 - 87571^2| = 35$.

P.S. You might want to consider using Python (PyPy3) for this problem, but it's really up to you :)

K4C

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

How could we not have a K4C problem :)

In this problem, the length of a string s is denoted by $|s|$.

Your task is to count the number of *cool* substrings in a given string s consisting of the characters `K4C`.

Consider any substring $s[l, r] := s_l s_{l+1} \dots s_r$ where $1 \le l \le r \le |s|$. Let cnt_X be the number of occurrences of X in $s[l, r]$. Then, the substring is said to be *cool* if it satisfies both of the following conditions:

1. $\text{cnt}_K, \text{cnt}_4, \text{cnt}_C \ge 1$
2. $\lfloor \frac{\text{cnt}_C}{\text{cnt}_K} \rfloor = 4$ †

† $\lfloor x \rfloor$ is x rounded down (e.g. $\lfloor 2.8 \rfloor = 2, \lfloor 5 \rfloor = 5$). You may also interpret this equation as *the result of integer division equals* 4.

Input

Each test contains multiple test cases. The first line of input contains a single integer T ($1 \le T \le 10^5$) — the number of test cases. The description of the test cases follows.

The first line contains a string s ($1 \le |s| \le 10^6$) consisting of the characters `K4C`.

It is guaranteed that the sum of $|s|$ over all test cases does not exceed 10^6.

Output

For each test case, output the number of cool substrings.

Scoring

This problem is composed of 4 subtasks:

Subtask	Additional Constraints	Points
1	$\|s\| \le 5$	5
2	$\sum \|s\| \le 5000$	15
3	`4` appears exactly once in s	40
4	None	40

Example

standard input	standard output
3 KC4 K4CCCC KCCCC4CCCCK	0 1 3

Note

In the first test case, there are no cool substrings.

In the second test case, the only cool substring is the string itself, `K4CCCC`.

In the third test case, there are 3 cool substrings:

1. $s[1,6] =$ `KCCCC4`
2. $s[6,11] =$ `4CCCCK`
3. $s[1,11] =$ `KCCCC4CCCCK`

Note that the substring `KCCCC` isn't cool as it doesn't satisfy the first condition ($\text{cnt}_4 = 0$). Similarly, `KCCCC4C` isn't cool as it doesn't satisfy the second condition ($\lfloor \frac{\text{cnt}_C}{\text{cnt}_K} \rfloor = 5$).

Because of the large input, it is recommended to use fast I/O for this problem to avoid unjustified TLEs. Feel free to use the attached templates in the "contest materials" section.

Math 101 (easy version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A palindrome is an integer that reads the same forwards and backwards in base 10. For example, $9, 101, 4444$ are palindromes, whereas $-1, 12, 8080$ are not.

Dr.D is the professor who teaches Math 101 at your uni. She's lazy and likes palindromes, so the midterm has only one question:

1. Given an integer k, find any palindromic multiple of k less than 10^{18}. (100 PTS)

You haven't come to class in weeks. Are you cooked?

Input

Each test contains multiple test cases. The first line of input contains a single integer T ($1 \le T \le 101$) — the number of test cases. The description of the test cases follows.

The first and only line contains an integer k ($1 \le k \le 101$).

Output

For each test case,

- If at least one solution exists, print *any* solution.
- Otherwise, print -1.

Scoring

This problem is composed of 3 subtasks:

Subtask	Add'l Constraints	Points
1	$k \le 9$	20
2	$k \le 20$	39
3	None	41

Example

standard input	standard output
4 2 27 37 101	2 999999999 12345678987654321 10201

Note

In the first test case, you can also print 22, but not 20 because it is not a palindrome ($20 \neq 02$).

In the second test case, 36963 is another acceptable answer.

Math 101 (hard version)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Continued from Math 101 (easy version) ...

Dr.D realized the midterm was too easy. To punish you all for skipping class, she decided to make the finals way harder. Again, there's only one question:

1. **Given an integer k, find ALL palindromic multiples of k from l to r, inclusive. (100 PTS)**

But let me tell you something. Dr.D will only check if the *number* of solutions is correct (shh...) So I'm sure you will do well!

Input

Each test contains multiple test cases. The first line of input contains a single integer T ($1 \le T \le 101$) — the number of test cases. The description of the test cases follows.

The first and only line contains three integers k, l and r ($1 \le k \le 101, 1 \le l \le r \le 10^{18}$) — the parameters of Dr.D's problem.

Output

For each test case, output the number of solutions on a separate line.

Scoring

This problem is composed of 5 subtasks:

Subtask	Additional Constraints	Points
1	$r - l \le 10^4$	15
2	$r \le 10^9$	15
3	Both l and r are powers of 10	30
4	$T = 1$	30
5	None	10

Example

standard input	standard output
3 2 8 80 11 44444444444444444 55555555555555555 81 1 1000000000	4 10101010 1

Note

In the first test case, there are 9 palindromes in the range $[8, 80]$. They are $8, 9, 11, 22, 33, 44, 55, 66, 77$. However, only $8, 22, 44, 66$ are divisible by $k = 2$, so the answer is 4.