The 7^{th} UIT ACM Programming Contest

May 14, 2017

Problems Overview

There are seven (7) problems in the packet, using letter A - G

Problem A - Binary String

Problem B - Combined Number

Problem C - Extension Evaluation

Problem D - Linux Girl

Problem E - Main Board

Problem F - Missing Bit

Problem G - Sheep and Wolves

*R*emember: For all problems, read the input data from standard input and write the results to the standard output

Problem A
BINARY STRING

Time Limit: *1 Second*

Let's start with a bunch of definitions.

We called a string X containing only letter '0' and '1' is a binary string, the K continuous letters of X is a K-substring of X. Each K-substring of X is identified by its location of starting letter in X. A K-unique binary string is a binary string which doesn't have two K-substring having same value.

For example, "1010" is a 3-unique binary string; "11111100001" is not a 3-unique binary string because it has four different 3-substrings started from first, second, third and fourth letter which has value "111".

Your task is finding the longest K-unique binary string.

Input

The input contains only one integer number K ($0 < K \leq 16$).

Output

Display in two lines are the length and the string of the longest K-unique binary string, using the smallest value (when you convert your binary string to binary number) if there is more than one result.

Sample

Sample Input	Sample Output
3	10 0001011100

Problem B
COMBINED NUMBER

Time Limit: *1 Second*

You are given two positive integers A and B. Known that these two integers are created from a positive integer X by deleting some digits from it. The deleting part with same digits order with X is combined to create A and the remaining part (also has same digits order with X)is combined to create B. Now, Your task is simply finding the smallest and largest number X could be.

Input

The input consists of two lines. Each line contains one positive integer with no more than 30 digits. The first line is represented for number A and the second line is represented for number B.

Output

Display the smallest and largest number X as described above.

Sample

Sample Input	Sample Output
1268 46235	124623568 462351268

Problem C
EXPRESSION EVALUATION

Time Limit: *1 Second*

Most of the programming languages used in practice perform strict evaluation. When a function is called, the expressions passed in the function arguments (for instance $a + b$ in $f(a + b, 3)$) are evaluated first, and the resulting values are passed to the function.

However, this is not the only way how expressions can be evaluated. In fact, even such a mainstream language as C++ is sometimes performing lazy evaluation: the operators && and || evaluate only those arguments that are needed to determine the value of the expression.

You are now working on a comparative study of the performance of the lazy and strict evaluation. You have to evaluate in both ways a set of expressions that follow this simplified syntax:

- An **expression** is either a constant, a name, or a function call
- A **constant** is a signed 32-bit integer; for example, 3, 0, -2, 123 are constants;
- A **name** is a name of a built-in or user-defined function, for example, f , or add are names; names are words containing up to 32 lowercase letters from the English alphabet;
- **function call** has the form: $(function\ arg1...argN)$, where function is an expression that evaluates to some function of N arguments, and $arg1...argN$ are expressions that evaluate to arguments passed to the function. For example, (f 3 5) , or (add 2 (add 1 2)) are valid function calls.

Expressions are evaluated according to the following simple rules:

- constants evaluate to themselves
- names evaluate to the functions they denote
- function calls:

  - **In lazy evaluation**: the first expression is evaluated first to obtain a function, whose function body, with formal parameters substituted for the expressions provided as the arguments, is evaluated; however, whenever any argument gets evaluated while evaluating the function body, the resulting value will replace all occurences of the same parameter in that function body. In other words, the expression passed in the argument is never evaluated more than once.
  - **In strict evaluation**: all expressions are evaluated first: the first expression should evaluate to a function, the remaining to values that are used as function arguments; the result is the result of evaluating the corresponding function body, where all occurences of formal parameters are replaced by the values obtained by evaluating the arguments.

The following built-in functions are available:

- $add\ x\ y$ - sum of the constants x and y
- $sub\ x\ y$ - returns the value $x - y$
- $mult\ x\ y$ - product of x and y
- $div\ x\ y$ - integer division
- $rem\ x\ y$ remainder (same semantics '%' in C, C++ or Java)
- $true\ x\ y$ - always returns x,
- $false\ x\ y$ - always returns y
- $eq\ x\ y$ - returns true if x and y is the same constant, otherwise returns false
- $gt\ x\ y$ - returns true if x is greater than y, otherwise returns false.

User-defined functions are defined using the following syntax:

$$function_name\ arg1...argN = body$$

where $arg1...argN$ are distinct words (up to 32 English lowercase letters), denoting the formal parameters of the function, and the body is an expression.

The formal parameters can occur in the body of the function in place of constants or names. The function name and the formal parameters are separated by a single space. There is one space on both sides of the "=". Functions with zero (no) arguments are legal. Note that the formal parameters can overshadow the function names (i.e. op in definition of not in sample input overshadows the function name op, but each function must have a unique name.

Input

The first part of the input contains (less than 1000) lines with one function definition each, followed by a single empty line. Forward references (that is, referring to functions defined later in the input) and recursion are legal.

The second part of the input contains less than 1000 test expressions. Each test expression is an expression occupying a single line. Function names and the arguments are always separated by a single space, but there are no extra spaces around parentheses (see sample input). There is an empty line after the last expression. Expressions are to be evaluated by both the lazy and the strict evaluation.

You can assume that all function definitions and expressions are syntactically correct, and that the arithmetic built-in functions (add, sub, mult, div, rem, eq, gt) will always be called with integers only, and no division by 0 occurs. Overflows outside the 32-bit integer range are legal and do not require any special treatment (just use the value produced by C, C++, or Java operators $+$, $-$, $*$, $/$, or $\%$). In strict evaluation, built-in functions evaluate all their arguments too. In lazy evaluation, arithmetic built-in functions always evaluate all their arguments. All lines on the input contain no more than 255 characters including spaces.

Output

Display a table in exactly the following format:

operator	lazy_evaluation	strict_evaluation
add	add_{lazy}	add_{strict}
sub	sub_{lazy}	sub_{strict}
mult	$mult_{lazy}$	$mult_{strict}$
div	div_{lazy}	div_{strict}
rem	rem_{lazy}	rem_{strict}

where each op_{lazy} is an integer - how many times op has been executed in lazy evaluation of all expressions, and op_{strict} is the number of evaluations of op in strict evaluation. Each column in the header row is separated by a single space, each column in other rows is separated by two tab character.

If the evaluation of a test expression does not terminate after a total of 2345 function evaluations, you can assume that it is in an infinite loop, the program should skip that expression, and do not count it into the totals (omit counting operations both in lazy and strict evaluation of this expression).

Sample

Sample Input

```
if cond truepart elsepart = (cond truepart elsepart)
fact x = (facta x 1)
facta x a = (if (eq x 0) a (facta (sub x 1) (mult a x)))
and x y = (x y false)
ident x = x
two = 2
op op x = ((if (eq op 1) add sub) op x)
not op = (op false true)
sum n = (suma n 0)
suma n a = (((gt n 1)
suma false) (sub n 1) (add a n))

(true (add 1 2) (mult 1 2))
5
true
(and (gt (op (sub 2 1) 1) 5) (eq (two) (op 1 1)))
(false (sub 1 2) (sum 4))
((eq (true 1 2) (false 2 1)) (add 1 2) (sub 1 2))
(fact 3)
```

Sample Output

```
operator lazy_evaluation strict_evaluation
add		7		8
sub		4		7
mult		0		1
div		0		0
rem		0		0
```

Problem D
LINUX GIRL
Time Limit: *1 Second*

The legendary of Linux Girl has been told around UIT years by years. The story started a long time ago, at the biggest class room of block C. There were five tables in the front row numbered by 1, 2, 3, 4 and 5 and table number 1 is the left-most in the row. Those tables belong to five beautiful female students named Hoa, Lan, Mai, Trang and Trinh (these names were given in alphabetical order only. So, do not jump to any premature assumption like Hoa is sitting in table 1 and such).

Although those girls studied at the same class, they are really much different from each other. They love different colors: Red, Orange, Yellow, Green and Blue (again, those colors was listed in the order of their wave-length, so do not make any assumption). They use laptops with different operating systems: ChromeOS, FreeBSD, Linux, MacOS and Windows. They buy clothes from different brand: D&G, Gucci, Hermes, L.V and Prada. And also they are proficient in different programming language: C++, Javascript, PHP, Python and Ruby.

A handsome rich man studied in that same classroom who wanted to find his perfect match. He does not care much about grade or skill, his main focus was to find a bride to live happily forever after. Now, with five beautiful girls, he want to find a really special girl, someone cool, delicate and thoughtful. He decided to choose the girl who use Linux. And now he's still alone.

After years, you decide to find the truth. All you have are some facts from the man and need to find who was the girl using Linux.

Input

The input starts with a line containing single positive integer N ($0 < N \leq 2000$), which is the number of facts you have gathered about given girls. Following that are N lines, each containing three words separated by 1 space between them using the following format (for the sake of simplicity, only lower case letters are used):

- The first and third word is one of only these keywords:

```
1  2  3  4  5
hoa  lan  mai  trang  trinh
```

```
red  orange  yellow  green  blue
chromeos  freebsd  linux  macos windows
d&g  gucci  hermes  l.v prada
c++  javascript  php  python ruby
```

- The second word represents the relationship between first and third word. We only have these relationships: **definite, left-of, right-of** and **next-to**
  - **definite** tells that the first and third fact words definite apply to the same girl.
    For example: "blue definite lan" tells that "the girl who love blue is definitely the girl name Lan"
  - **left-of** tells that the first fact word applies to the girl immediately to the left of the one which the third fact word applies.
    For example: "hoa left-of ruby" means that "The girl name Hoa sits immediately to the left of the girl whose program is in Ruby"
  - **right-of** tells that the first fact word applies to the girl immediately to the right of one which the third fact word applies.
    For example: "trinh right-of 3" means "The girl named Trinh sit immediately to the right of the girl sitting in table number 3"
  - **next-to** tells that the two fact words apply to girls sit next to each other.
    For example: "prada next-to linux" means that "the girl wear Prada sit in the next table (either to the left or the right) of the girl who use Linux"

You may assume that there will be no inconsistencies in the input data. In other words, at least one girl may use the Linux operating system without violating any of the constraints.

Output

Display "*N* uses linux", with *N* be replaced by hoa, lan, mai, trang, or trinh. if you can safely determine who uses Linux. Otherwise, display "cannot identify linux user".

Sample

Sample Input	Sample Output
8 mai left-of linux mai left-of 4 c++ definite 1 d&g definite 1 yellow definite 1 chromeos definite yellow windows next-to chromeos macos left-of prada	cannot identify linux user
8 yellow definite 1 d&g definite 1 ruby definite 5 freebsd definite 2 chromeos definite 3 macos definite 4 windows definite 5 hoa definite 1	hoa uses linux

Problem E
MAIN BOARD

Time Limit: *1 Second*

UIT microchip team has introduced their new main board. The main board is divided by N sectors indexed by letter A to Z. Each sector has their feature function and could affect other sectors. The special of this board is its auto-recovery feature. After a long time unused, all its sector will be turned to sleeping mode. Only some core sectors M still keep enable to recovery the main board anytime. After you turn on core sectors M, for each interval time T, sectors V will be turned on again if all sectors directly affecting to its are on (including core sectors).

Now, you are hired to control the quality of new main board. Your task is finding which main board could be full enable again (all sectors are enable) or not.

Input

The Input contains several lines. The first line includes single integer $N(1 < N \leq 26)$ is the number of sector. the second line includes a positive integer K, the number of relation between sectors. The third line contains a string has $M(1 \leq M \leq 3)$ letters, which is the name of core sectors. The next K lines, each line contains 2 letters C_1 and C_2, representing for sector C_1 could directly affect sector C_2.

Output

Display how many interval T times needed to full enable main board from sleeping mode. If the main board cannot fully recover, display -1.

Sample

Sample Input	Sample Output
6 11 HAB AB AC AH BD BC BF CD CF DF FH HC	3

Problem F
MISSING BIT

Time Limit: *1 Second*

Breaking News! The UIT's Security Network team is planning to introduce their new transfer system. According to a team member, their new system will send data as a list of 255-bit numbers each time calling the origin numbers. For each origin number in list, before transfering, the transfer system will generate a random number K and execute K times of bit reversal calculation (replace bit 0 by bit 1 and bit 1 by bit 0) on some random bits, one bit could be reverse more than one time.

Unfortunately, an hardware issue has caused with transfer system. Somehow, after transfer process, one bit will be lost on each received number. It will take a long time to fix this on system hardware. So, you are a supper cool software engineering. On the meaning time, you can find out what is the value of the missing bit in received number using system log information. Known that system log can only gives you the information about number of bit 1 in origin number (before bit reversal calculation) and number of bit 1 in received data (with one missing bit)

Input

The input begins with a line containing one integer N ($0 < N \leq 10$), which is the number of bit numbers in your list. Following this are N lines, line i^{th} describes the transfer process of bit number i^{th} in list as three integers P, K, Q, where P is the number of bit 1 in origin number, K is the number of reversal calculation ($1 \leq K \leq 8.10^6$), Q is the number of bit 1 after transfer with one missing bit.

Output

Display a N-bit number, with bit i^{th} is the missing bit of bit number i^{th} corresponding with input.

Sample

Sample Input	Sample Output

2
208 6 212
191 104 56
	01

Problem G
SHEEP AND WOLVES

Time Limit: *1 Second*

Let's consider a problem known as the Sheep and Wolves 2.0.

A man lives on the east side of a sheep-walk. He wishes to bring his sheep swarm to a village on the west side of the river to protect them from an impending wolf attack. The distance between two locations is S. He could also transport them by a truck with constant velocity V_t or let them walking by themselves (yes, they are walkable) with constant velocity V_s. However, his truck is only big enough to hold himself and four sheep more. He needs to transport all sheep in the fastest way because the wolfs is nearby.

How does the man solve his problem?

Input

The input consists of a single line containing one positive integer N ($0 < N \leq 1000$), which is the number of sheep, and three positive floating-point numbers no larger than 100 S, V_t, V_s ($V_s < V_t$) representing the road distance, the constant velocity of transport by truck and constant velocity of walking.

Output

Display the minimum time to bring all sheep to the village. Your answer should have an absolute or ground up to thousandths place.

Sample

Sample Input	Sample Output
10 20 5 1	10.400

This is the end of the packet. Good Luck !