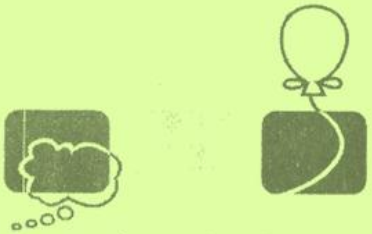


IBM event sponsor

ACM-ICPC Thailand Southern Area Programming Contest 2010

13 July 2010

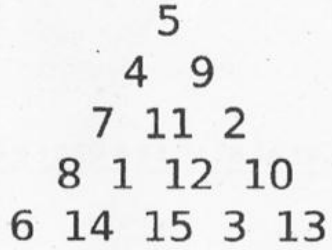
Hosted by Prince of Songkla University, Phuket Campus

- There are **8 problems** (A-H) to solve within 210 minutes (3.5 hours).
- Solve as many problems as you can, in an order of your choice.
- Use C or C++ or Java to program at your convenience for any problems.
- Input and output of each program are **standard input** and **output**.

Problem A:	**Absolute-Difference Triangle Puzzle**
Problem B:	**Propositional Satisfiability**
Problem C:	**Game – Mouse and Cheese**
Problem D:	**Text Messaging**
Problem E:	**Meeting Schedule Arrangement**
Problem F:	**Getting a good bonus!**
Problem G:	**Synergy**
Problem H:	**Paul, The World Cup Predictor**

Problem A
Absolute-Difference Triangle Puzzle

An Absolute-difference triangle is a geometric arrangement of numbers in a triangle form. An Absolute-difference triangle with order-*n* is a triangle with side of *n*, where the number at each apex (of any smallest sub-triangles) is a difference of the two numbers on its base. (Smallest triangle is a triangle with order-*2*). See the given figure.



Normally, the numbers in Absolute-difference triangle are a set of consecutive integer starting from 1. Some previous works prove that the maximum order of the Absolute-difference triangle, using this set of consecutive integers, is order-*5* (as in the figure shown above). Anyway to solve our puzzle, a given set of number will be provided which may extends the highest order of the triangle.

Goal

To build all possible triangles for each set of input number.

Input

Input is a standard input which contains a sequence of puzzles. There are no more than 300 puzzles for an input. Each puzzle is within 2 lines.

- The first line is the order of the triangle.
  - The order is an integer number more than 1 and less than 7.
- The second line is a set of number where the triangle must contains with.
  - Each number is an integer delimited by a space.
  - The valid range is from 1 to 100.
  - Each number is also unique.
- The blank line is the termination of the input.

Output

For each input puzzle, write 2 parts of output as follows

- In the first line, write the total of number of solutions (no. of triangles which can be created).
- In the following lines, write each solution in each line. Each solution is a sequence of number in the triangle starting from top to bottom and from left to right. The solutions must be sorted ascending according to lexicographical order. If there is no solution, write 0 (zero).

Sample Input

```
2
2 1 3
5
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2
5 15 80
7
5 49 4 24 12 60 2 23 1 20 36 43 29 9 15 17 8 54 72 28 16 19 3 6 51 7 18 45
```

Sample Output

```
4
1 2 3
1 3 2
2 1 3
2 3 1
2
5 4 9 7 11 2 8 1 12 10 6 14 15 3 13
5 9 4 2 11 7 10 12 1 8 13 3 15 14 6
0
2
4 2 6 3 1 7 19 16 17 24 9 28 12 29 5 45 36 8 20 49 54 60 15 51 43 23 72 18
4 6 2 7 1 3 24 17 16 19 5 29 12 28 9 54 49 20 8 36 45 18 72 23 43 51 15 60
```

More Explanations

There are 4 puzzles in this sample input.

- For the first test case, the order of the triangle is 2, and the given set of number is 2, 1 and 3. The output has 4 solutions, which are listed according to their lexicographical order.
- The second case, the order is 5. The given set of number is a consecutive integer starting from 1 to 15. Two solutions are found and listed.
- The third case, no solution can be found.
- For the forth case, the order is 7. Two solutions can be found and listed.



event sponsor

B

Problem B
Propositional Satisfiability

A ***Minimal Propositional Formula*** (MPF) is a string of symbols made from propositional atoms using the boolean connectives & ("and"), + ("or"), ! ("not"), and brackets, in the appropriate way. More accurately:

1. Any propositional atom (a, b, c, . . . , z) is an MPF.
2. If A is an MPF then so is (!A).
3. If A, B are MPFs then so are (A & B) and (A + B).
4. That's it: nothing is an MPF unless built by these rules.

Brackets can be omitted. To get rid of brackets, we order the boolean connectives according to decreasing binding strength:

(strongest) !, &, + (weakest)

This is like in arithmetic, where × is stronger than +. This means that 2 + 3 × 4 is read as 2 + (3 × 4), not as (2 + 3) × 4. So:

The semantics (meaning) of an MPF is simply given by the following truth tables:

p + q & r **is read as** p + (q & r), not as (p + q) & r.

!p & q **is read as** (!p) & q, not as !(p & q).

p	q	p & q	p + q	!p
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE

An MPF is *satisfiable* if there is a *situation* that the MPF is evaluated to be TRUE. If there is no such situation, the MPF is *unsatisfiable*. For examples:

p & q is satisfiable on the situation that both p and q are TRUE.

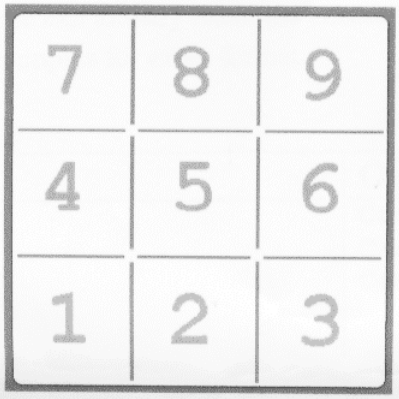
p & !p is unsatisfiable as there is no situation that can make it to be TRUE.

Your job is to write a program to take an ***MPF as an input*** (from standard input) and determine whether is satisfiable or not, by ***outputting YES or NO*** (to standard output) accordingly. The maximum length of the input string is 100 characters including spaces and brackets.

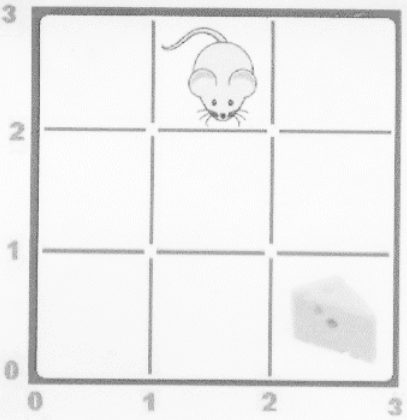
Sample Input	Sample Output
a q + !q p & q + r & (s + !t) a & (b + !c) & (d + f + g) & !(d + f + g) p & !p	YES YES YES NO NO

Problem C
Game – Mouse and Cheese

SOHA and TARA have recently invented a new game called "Mouse and Cheese". As the name suggests, this game involves a mouse searching for a piece of cheese. The game is played on a 3x3 board as shown in the diagram below. Each cell is uniquely numbered with an integer between 1 and 9. The board contains 12 sticks (the blue lines in the diagram).



Initially a mouse and a piece of cheese are placed on two different cells. In the example to the right, the mouse is placed on 'cell 8' and the cheese is placed on 'cell 3'.



The rule of the game goes like this: It's a two-player game where the players make moves alternately. SOHA, being player 1, goes first. In each move, the player selects a stick and throws it away. After each move, if the mouse can reach the cell containing the cheese without touching any of the remaining sticks, then that player is declared as the winner. If both play perfectly, who wins?

Before the game starts, some of the sticks are removed. You will be given the coordinates of these sticks and the cell numbers of the mouse and cheese. A stick is represented using the coordinates of its end points. The lower left of the grid is the origin (0, 0). The two sticks surrounding the cheese, in the example above, has coordinates "2 0 2 1" and "2 1 3 1". Note that the end-points of a stick can be given in any order. That is "2 0 2 1" could be given as "2 1 2 0".

Input

The first line of input is an integer **T**(**T**<1000) that determines the number of test cases. Each case starts with 3 integers **S**, **C** and **R**. **S** and **C** denotes the location of the mouse and cheese respectively. Each of the next **R** lines contains 4 space-separated integers that give the coordinates of the sticks removed from the board before the start of the game.

Notes and Constraints

```
1 <= S,C <= 9
S != C
0 <= R <= 12
```

For each case, all the removed sticks will be distinct.

Output

For each case, output the case number first, starting with 1, followed by the name of the player who wins. If the mouse can already reach the cheese before the game starts, output "No Cheese!" instead. Look at the samples for exact format.

Sample Input	Output for Sample Input
3 1 2 1 1 1 1 0 1 4 0 7 2 2 1 2 1 3 2 2 2 3	Case 1: No Cheese! Case 2: SOHA Case 3: TARA

Hint: game theory, trivial BFS.

Problem D
Text Messaging

You can type alphabets on a cell phone using numeric keypads using the following key mapping.

Numbers	Alphabets
2	abc
3	def
4	ghi
5	jkl

Numbers	Alphabets
6	mno
7	pqrs
8	tuv
9	wxyz

Since one numeric keypad represents many alphabets, on the standard system, the user must press the keypad many times to choose the right alphabet. For example to type the word "rat", the user must press 77728. (777 to get "r", 2 to get "a", and 8 to get "t".)

After the word prediction software, the user does not have to do that anymore. To type "rat" the user can type 728, and the prediction software would guess "rat" automatically. For another example, consider the word "cat." To type that the user needs only press 228. We call the key sequence required to type a given word as its *key sequence*. Thus, the key sequence of "cat" is 228, and the key sequence of "rat" is 728.

However, this is not perfect because many words share the same key sequence. For example, "cat" and "bat" have the same key sequence 228.

Given a set of words, your task is to find out how many *different* key sequences needed to type any words in the set. For example if the set is {"rat", "cat", "bat"}, the number of different key sequences is 2, i.e., 228 and 728.

Input

The first line of the input contains an integer T ($1 <= T <= 6$) denoting the number of test cases. After that T test cases follow.

The first line of each test case contains an integer N ($1 <= N <= 1{,}000$) denoting the number of words. After that N lines follows. Each line contains different word. Each word consists of only small English alphabets, and the maximum length of any words is at most 20 characters.

Output

For any test case, your program should output an integer K denoting the number of different key sequences.

Example

Sample Input	Sample Output
2 3 cat rat bat 5 abcdefg cccdddi aaafefh hello world	2 3

Explanation

In the first case, two sequences are 228 and 728. In the second case, the three sequences are 2223334, 43556, and 96753.

Problem E

Meeting Schedule Arrangement

Mr. Trasapong, the head of Computer Engineering Department, would like to have a meeting as soon as possible to address some issues of the department. However, other members in his department are so busy that it is very difficult to arrange their schedules for the meeting. So, in order to settle the meeting date, he requested every member to send back a list of convenient dates by E-mail. Your mission is to help him by writing a program that chooses the best date from the submitted lists. Your program should find the most convenient date for the most of the members. If there is more than one such day, the earliest is the best.

Input

The input has multiple data sets, each starting with a line containing the number of members in the department and the quorum* of the meeting.

N Q

Here, *N*, meaning the number of the members in the department, and *Q* meaning the quorum, are positive integers. *N* is less than 20, and, of course, *Q* is less than or equal to *N*.

N lines follow, each describing convenient dates for a member in the following format.

M $Date_1$ $Date_2$ ... $Date_M$

Here, *M* means the number of convenient dates for the member, which is an integer greater than or equal to zero. The remaining items in the line are his/her dates of convenience, which are positive integers less than 31, that is, 1 means tomorrow, 2 means the day after tomorrow, and so on. They are in ascending order without any repetition and separated by a space character. Lines have neither leading nor trailing spaces.

A line containing two zeros indicates the end of the input.

Output

For each data set, print a single line containing the date number convenient for the largest number of members. If there is more than one such date, print the earliest. However, if no dates are convenient for more than or equal to the quorum number of members, print 0 instead.

* Note: quorum means the smallest number of the people who must present at a meeting.

Sample Input	Sample Output
3 2 2 1 4 0 3 3 4 8 3 2 4 1 5 8 9 3 2 5 9 5 2 4 5 7 9 3 3 2 1 4 3 2 5 9 2 2 4 3 3 2 1 2 3 1 2 9 2 2 4 0 0	4 5 0 2

Problem F
Getting a good bonus!

Edward is the project manager of Cha-Cha-Cha (Sam-Cha) Software Company International gets many software projects to develop from SIPA. Each software project carries one score, plus a bonus if submitted within a specified number of weeks. The deadline to get the bonus and the number of bonus scores are different for each project. For example, if a project has a deadline of 6 weeks and carries bonus 10 scores, then it earns 10 bonus scores if it is submitted before the end of the 6^{th} week. Each project takes exactly one week to complete. For instance, suppose there are seven projects with deadlines and bonuses as follows:

Project number	1	2	3	4	5	6	7
Deadline for bonus	1	1	3	3	2	2	6
Bonus score	6	7	3	1	4	5	1

The maximum bonus score is 16, which can be achieved by completing the projects in the sequence 2,6,3,1,7,5,4.

Note that there are also other sequences that achieve the same score. Your task is to find a schedule to complete all software projects so as to maximize bonus score.

Input

The first line contains an integer **N** ($1 \le \mathbf{N} \le 50$) which determines number of the test cases. The following line is to indicate **M** ($1 \le \mathbf{M} \le 24$), where **M** is the number of software projects. This is followed by **M** lines, each containing two integers. The first integer is the deadline for the i^{th} project and the second integer is the bonus score assigned to the i^{th} projects, ($1 \le i \le$ M)

Output

For each test case, print out the maximum score that can be obtained.

Sample Input	Sample Output
2 7 1 6 1 7 3 2 3 1 2 4 2 5 6 1 4 2 10 1 9 2 7 7 1	15 20

Problem G

Synergy (รวมพลัง)

ในห้องปฏิบัติการทางวิทยาศาสตร์แห่งหนึ่ง นักวิทยาศาสตร์ได้ค้นพบสิ่งมีชีวิตขนาดจิ๋วชนิดหนึ่ง และได้ตั้งชื่อไว้เรียกสิ่งมีชีวิตเหล่านี้ไว้อย่างชั่วคราวว่า **"รวมพลัง"** นักวิทยาศาสตร์ยังไม่รู้แน่ชัดเกี่ยวกับคุณลักษณะต่าง ๆ ของมัน แต่จากการเฝ้าสังเกตุ นักวิทยาศาสตร์ค้นพบลักษณะสำคัญบางประการ ดังนี้

- *รวมพลัง* สามารถแบ่งออกเป็นหลายชนิด
- เมื่อตอนเกิด *รวมพลัง* แต่ละตัวจะมีช่วงชีวิตเท่ากันหมด คือ 1 หน่วย
- *รวมพลัง* สองตัวหรือหนึ่งคู่จะสามารถรวมตัวกันเกิดเป็น *รวมพลัง* ตัวใหม่ได้ และ*รวมพลัง* ตัวใหม่นั้นจะมีช่วงชีวิตที่ยาวนานมากขึ้น โดยช่วงชีวิตใหม่นี้จะคำนวณได้จากผลรวมของช่วงชีวิตของ *รวมพลัง* ตัวเก่าทั้งสองคูณกับค่าคงที่ค่าหนึ่ง ซึ่งการรวมกันของแต่ละคู่นั้นจะเป็นไปตามกฎที่นักวิทยาศาสตร์ค้นพบดังในตารางต่อไปนี้

กฎที่	ชนิดใหม่	ชนิดตั้งต้นตัวที่ 1	ชนิดตั้งต้นตัวที่ 2	ค่าคงที่การคูณ
1	Aa	a	a	1
2	AA	a	a	17
3	bb	b	b	3
4	x	aa	bb	2
5	x	x	a	2
6	c	c	a	3

- และเมื่อนักวิทยาศาสตร์ทำการจัดวาง *รวมพลัง* ที่เกิดใหม่หลายตัวติดเข้าด้วยกันเป็นสาย *รวมพลัง* แต่ละตัวในสายจะพยายามจับคู่กับตัวข้างเคียง เพื่อรวมกันเข้าเป็นชนิดใหม่ที่มีช่วงชีวิตที่ยาวนานขึ้น และตัวใหม่ที่เกิดขึ้นก็จะพยายามจับคู่กับตัวข้างเคียงอีกต่อไป ไม่ว่าตัวข้างเคียงนั้นจะเป็นตัวที่เกิดใหม่ หรือเป็นตัวที่เกิดจากการรวมกันมาแล้วก็ตาม ซึ่งทำให้การรวมกันสำหรับแต่ละสายนั้น อาจเป็นไปได้หลากหลายรูปแบบ (แตกต่างกันไปขึ้นอยู่กับทั้งลำดับในการรวมและกฎที่ใช้)

เพื่อเป็นตัวอย่าง รูปแบบในการรวมบางแบบได้แสดงไว้ในตารางต่อไปนี้

ขั้นตอนที่	สาย	ช่วงชีวิต	กฎ	หมายเหตุ
1	a a b b	1 1 1 1	-	สายที่เกิดใหม่
2	aa bb	2 6	1, 3	
3	x	16	4	รวมกันจนสิ้นสุด และรวมได้ทั้งหมด
1	a a b b	1 1 1 1	-	สายตั้งต้น
2	AA bb	34 6	2, 3	รวมกันจนสิ้นสุด แต่รวมได้ไม่หมด
1	a c a c	1 1 1 1	-	สายตั้งต้น
2	c c	6 6	6, 6	รวมกันจนสิ้นสุด แต่รวมได้ไม่หมด
1	a c a c	1 1 1 1	-	สายตั้งต้น
2	a c c	1 6 1	-, 6, -	
3	c c			รวมกันจนสิ้นสุด แต่รวมได้ไม่หมด

ในตัวอย่างแรก ในขั้นตอนแรก จะมี *รวมพลัง* ที่เกิดใหม่อยู่สาย 4 ตัว คือ "a a b b" โดยจะมีช่วงชีวิตคือ "1 1 1 1" ตามลำดับ ในขั้นที่สอง จะเกิดการรวมตัวเข้าด้วยกันของรวมพลัง 2 คู่ในสาย คือ "aa(a+a)" และ "bb(b+b)" เกิดเป็นสาย "aa bb" ที่มี *รวมพลัง* อยู่ 2 ตัว โดยใช้กฎที่ 1 และ 3 ตามลำดับ โดยช่วงชีวิตของ aa จะเป็น 2=(1+1)*1 และช่วงชีวิตของ bb จะเป็น 6=(1+1)*3 ในขั้นตอนที่สาม aa และ bb จะรวมกันเข้าเป็น x โดยใช้กฎที่ 4 และช่วงชีวิตจะเป็น 16=(6+2)*2

ในตัวอย่างที่สอง สายของ *รวมพลัง* ตั้งต้นจะเหมือนกันกับในตัวอย่างแรก แต่การรวมในขั้นตอนที่สอง จะใช้กฎที่ 2 และ 3แทน (ในตัวอย่างแรกใช้กฎที่ 1 และ3) ซึ่งจะทำให้เกิดสาย "AA bb" ที่มีช่วงชีวิตเป็น 34=(1+1)*17 และ 6=(1+1)*3 ตามลำดับ ซึ่งสายที่ได้นี้แม้จะเป็นสายที่รวมกันจนสิ้นสุดหรือไม่สามารถรวมต่อไปได้อีก แต่ก็ไม่สามารถรวมได้หมดทุกตัว ยังเหลือ *รวมพลัง* ในสายมากกว่าหนึ่งตัว อย่างไรก็ตาม หากทำการเปรียบเทียบกับการรวมกันในตัวอย่างก่อนหน้านี้จะพบว่า การรวมกันในตัวอย่างนี้จะก่อให้เกิด *รวมพลัง* ที่มีช่วงชีวิตยาวนานกว่า (34 เทียบกับ 16)

ในตัวอย่างที่สามและสี่ การรวมตัวกันนั้นจะเป็นไปตามกฎที่ 6 โดยในตัวอย่างที่ 3 จะเกิดการรวมกันสองคู่ที่เกิดจากลำดับที่ 1รวมกับ 2 และลำดับที่ 3 รวมกับ 4 ส่วนในตัวอย่างอย่างที่ 4 จะเกิดขึ้นจากลำดับที่ 2 รวมกับ 3 ซึ่งแสดงให้เห็นว่า กฎในการรวมตัวกันนั้น ขอเพียงให้ *รวมพลัง* ตั้งต้นทั้งสองมีชนิดตรงกันกับกฎโดยไม่สนใจลำดับ กล่าวคือ ลำดับจะเป็น a c หรือ c a ก็สามารถใช้กฎที่ 6 ได้เช่นกัน

Goal

นักวิทยาศาสตร์ต้องการให้คุณสร้างโปรแกรมเพื่อหาว่าการรวมตัวกันแบบใดจะก่อให้เกิด "***รวมพลัง***" ที่มีช่วงชีวิตยาวนานที่สุด จากสายตั้งต้นของ ***รวมพลัง*** ที่เกิดใหม่ พึงระลึกว่าคำตอบที่ได้ไม่จำเป็นต้องมาจากการรวมกันจนหมด หรือรวมกันจนสิ้นสุด (ในตัวอย่างที่สองแม้รวมกันได้ไม่หมด แต่ก็มี ***รวมพลัง*** ที่มีช่วงชีวิตยาวนานกว่าในตัวอย่างที่หนึ่ง)

Input

อินพุตเป็นอินพุตมาตรฐาน (standard input) ซึ่งประกอบไปด้วยข้อมูล 2 ส่วน และแบ่งแต่ละส่วนด้วยบรรทัดว่าง 1 บรรทัด

ส่วนแรกจะเป็นกฎในการรวม

- กฎแต่ละกฎจะอยู่ใน 1 บรรทัด
- แต่ละกฎจะประกอบด้วยข้อมูล 4 ส่วนย่อย ซึ่งแบ่งแต่ละส่วนย่อยด้วยช่องว่างอย่างน้อย 1 ช่อง
- ในส่วนย่อย 3 ส่วนแรกของแต่ละกฎ จะเป็นชนิดของ *รวมพลัง* โดยส่วนที่หนึ่งจะเป็นชนิดใหม่ ส่วนที่สองจะเป็นชนิดตั้งต้นตัวที่ 1 และส่วนที่สามจะเป็น ชนิดตั้งต้นตัวที่ 2 และชนิดทั้งสามชนิดนี้จะกำหนดโดยอักษรภาษาอังกฤษทั้งตัวใหญ่และตัวเล็กรวมทั้งตัวเลข(alphanumeric) โดยชนิดแต่ละชนิดจะมีความยาวไม่เกิน 20 ตัวอักษร
- ในส่วนย่อยสุดท้ายของแต่ละกฎ จะเป็นค่าคงที่การคูณ โดยค่านี้จะเป็นเลขจำนวนเต็มบวกตั้งแต่ 1 ถึง 100

ในส่วนที่สองจะเป็นคำถามจากนักวิทยาศาสตร์ที่ประกอบจากสายของ *รวมพลัง* ที่เกิดใหม่หลายสาย

- แต่ละสายของ *รวมพลัง* จะอยู่ใน 1 บรรทัด
- สายของ *รวมพลัง* จะประกอบด้วยลำดับของ *รวมพลัง* โดยแบ่งแต่ละลำดับด้วยช่องว่างอย่างน้อย 1 ช่อง
- *รวมพลัง* แต่ละลำดับจะกำหนดโดยอักษรภาษาอังกฤษทั้งตัวใหญ่และตัวเล็ก และมีความยาวไม่เกิด 20 ตัวอักษร (เช่นเดียวกันกับชนิดตัวตั้งต้นและชนิดใหม่ในส่วนแรก)

บรรทัดว่างหลังจากส่วนที่สองจะเป็นตัวบอกจุดสิ้นสุดของอินพุต

Output

สำหรับแต่ละสายของ *รวมพลัง* ตั้งต้น (อินพุตส่วนที่สอง) ให้สร้างผลลัพธ์ดังนี้

- ในบรรทัดแรก ให้เขียนจำนวนของ *รวมพลัง* ที่เกิดจากการรวมและมีช่วงอายุมากที่สุด ตามด้วยช่องว่างหนึ่งช่อง และช่วงอายุดังกล่าว

- ในบรรทัดต่อ ๆ ไปให้เขียนผลลัพธ์แต่ละผลลัพธ์ที่ทำให้เกิด *รวมพลัง* ที่มีช่วงอายุมากที่สุดดังกล่าว โดยแต่ละผลลัพธ์จะประกอบด้วยสามส่วนย่อย ซึ่งคั่นแต่ละส่วนย่อยด้วยช่องว่าง 1 ช่อง โดยแต่ละส่วนจะเป็นดังต่อไปนี้

    1. นิพจน์ที่แสดงการรวม

        - โดยนิพจน์นี้จะประกอบด้วยพจน์และหรือนิพจน์ (term and expression) ย่อย 3 ส่วนและมีรูปแบบดังนี้คือ

            $$term_1(term_2+term_3)$$

        - พจน์ที่หนึ่งจะเป็นชนิดของ *รวมพลัง* ชนิดใหม่ที่รวมได้
        - พจน์ที่สอง และสาม (โดยพจน์ที่ สองและสามเรียงกันตามลำดับที่ปรากฏในสาย) อาจเป็นหนึ่งในสองอย่างดังนี้
            - ชนิดของ *รวมพลัง* ตั้งต้น ทั้งสองตัว
            - นิพจน์ (ที่เกิดจากการรวมกันก่อนหน้า)

    2. ตำแหน่งแรกที่เกิดการรวม
    3. ตำแหน่งสุดท้ายที่เกิดการรวม

- ในกรณีที่มี *รวมพลัง* ที่มีช่วงชีวิตยาวที่สุดมากกว่าหนึ่งตัว ให้เรียงลำดับการเขียนแสดงผลแต่ละตัวโดยเรียงตามตำแหน่งแรกที่เกิดการรวม ตามด้วยตำแหน่งสุดท้ายที่เกิดการรวม และตามด้วยนิพจน์ตามลำดับ โดยเป็นการเรียงจากมากไปหาน้อย และสำหรับการเรียงตามนิพจน์ ให้เรียงตามลำดับแบบทีละอักษรตามรหัส แอส-กี้ (ASCII lexicographical order)

ตัวอย่างของ input และ output จะเป็นดังรูปข้างล่าง

Input ตัวอย่าง	Output ตัวอย่าง
aa a a 1 AA a a 17 bb b b 3 x aa bb 2 x x a 2 c c a 3 a a b b a a b b a a a b b a a a c a c a c	1 34 AA(a+a) 1 2 2 34 AA(a+a) 1 2 x(x(aa(a+a)+bb(b+b))+a) 1 5 2 70 x(a+x(a+x(bb(b+b)+aa(a+a)))) 1 6 x(x(x(aa(a+a)+bb(b+b))+a)+a) 1 6 1 6 c(a+c) 1 2 2 21 c(a+c(c+a)) 1 3 c(c(a+c)+a) 1 3

คำอธิบายตัวอย่าง

ในอินพุตตัวอย่าง จะมีกฎทั้งหมด 6 กฎ และมีสายตั้งต้นของ *รวมพลัง* ทั้งหมด 5 สาย คำอธิบายผลลัพธ์ของแต่ละสายจะเป็นดังนี้

ในตัวอย่างแรก จากสาย “a a b b” จะมีผลลัพธ์ทั้งหมด 1 ผลลัพธ์ และมีค่าช่วงชีวิตมากสุดอยู่ที่ 34 โดยผลลัพธ์ดังกล่าว คือ “AA(a+a) 1 2” หมายความว่า AA จะเป็น *รวมพลัง* ที่ได้ เกิดจากการรวม a เข้ากับ a และการรวมนี้จะเกิดจากตำแหน่งที่ 1 จนถึง 2 ซึ่งในตัวอย่างนี้จะไม่แสดงการรวมที่เป็นไปได้อีกแบบ แต่มีค่าช่วงชีวิตต่ำกว่า นั่นคือ “x(aa(a+a)+bb(b+b))” ซึ่งจะมีช่วงชีวิตเพียง 16

ในตัวอย่างที่สอง จากสาย “a a b b a”จะมีผลลัพธ์ 2 และมีช่วงชีวิตมากสุดอยู่ที่ 34 คือ “AA(a+a) 1 2” และ “x(x(aa(a+a)+bb(b+b))+a) 1 5” ซึ่งเกิดจากการรวม *รวมพลัง* จากตำแหน่งที่ 1 ถึง 2 และตำแหน่งที่ 1 ถึง 5 ตามลำดับ

ในตัวอย่างที่สาม จะได้ผล *รวมพลัง* ตัวสุดท้ายเป็น x ที่เกิดจากการรวม *รวมพลัง* ทุกตัวในสาย แต่การรวมสามารถสร้างได้จาก 2 รูปแบบ คือ x(a+x(a+x(bb(b+b)+aa(a+a)))) และ x(x(x(aa(a+a)+bb(b+b))+a)+a) (กรุณาสังเกตุการเรียงลำดับการเขียนแสดงผลลัพธ์ด้วย)

ในตัวอย่างที่สี่และห้า แสดงให้เห็นการรวมกันตามกฎที่ 6 “c c a 3” ซึ่งในตัวอย่างที่สี่จะใช้กฎนี้ได้เพียงที่ตำแหน่ง 1 ถึง 2 แต่ในตัวอย่างที่ห้า จะสามารถใช้กฎนี้ได้ทั้งในตำแหน่ง 1 ถึง 2 หรือ 2 ถึง 3 หรือ 3 ถึง 4 (แต่จากตัวอย่าง ผลลัพธ์ที่ดีที่สุดจะเกิดจากการใช้กฎนี้ในตำแหน่งที่ 1 ถึง 2 และ 2 ถึง 3)

H

Problem H
Paul, The World Cup Predictor

Paul (หรือ หมึกพอล) นักพยากรณ์ผลการแข่งขันชั้นเซียนของการแข่งขันฟุตบอลโลกผู้ให้ผลการแข่งขันแม่นยำจนทั่วโลกต่างสงสัยในความสามารถที่ทำนายผลการแข่งขันได้อย่างถูกต้องทุกครั้งในฟุตบอลโลก 2010 ที่ผ่านมา วันหนึ่ง Paul ได้เฉลยที่มาของความแม่นยำที่ผ่านมา นั่นคือการเก็บสถิติผลการแข่งขันไว้เพื่อใช้ในการประมวลผลแล้วนำมาใช้ในการวิเคราะห์อย่างง่ายด้วยการนับจำนวนครั้งในการชนะ แพ้ หรือเสมอ ผลการแข่งขันรูปแบบใดที่มีค่าสูงสุดก็จะให้คำตอบออกมาตามค่านั้น โดยมีหลักเกณฑ์การคำนวนดังนี้

- ถ้าผลการแข่งขันเดิมคือ ***ชนะ***ให้คิดค่าคะแนนเป็น *1*
- ถ้าผลการแข่งขันเดิมคือ ***แพ้*** ให้คิดค่าคะแนนเป็น *-1*
- ถ้าผลการแข่งขันเดิมคือ ***เสมอ***ให้คิดค่าคะแนนเป็น ***0***

หลังจากนั้นหาผลรวมค่าคะแนนทั้งหมด วิธีการวิเคราะห์คือ ถ้าผลคะแนนที่ออกมามีค่าเป็นบวก จะทำนายว่าทีมจะชนะ ถ้าเป็นค่าลบจะทำนายว่าทีมนั้นแพ้ หรือถ้ามีค่าเท่ากับศูนย์จะทายผลว่าจะเสมอกับทีมคู่แข่ง Paul ต้องการหาผู้ช่วยในการเขียนโปรแกรมในการนับผลการแข่งขันเพื่อใช้ในการทำนายในฟุตบอลโลกครั้งต่อไปที่บราซิลในปี 2014 จึงเป็นหน้าที่ของนิสิต นักศึกษาในภาคเหนือและภาคใต้ของประเทศไทยที่จะต้องช่วย Paul ในการทำนายผลการแข่งขันจากข้อมูลสถิติที่เก็บไว้

Input

บรรทัดแรกเป็นจำนวนการแข่งขันที่ต้องการทำนาย แต่ละการแข่งขันประกอบด้วยบรรทัดแรกมีชื่อทีมที่แข่งประกอบไปด้วย 2 ทีม และบรรทัดถัดไปเป็นผลการแข่งขันแต่ละครั้งที่ผ่านมา โดยตัวเลขแรกเป็นการได้ประตูของทีมแรกและตัวเลขที่สองเป็นการได้ประตูของทีมที่สอง

Output

วิธีแสดงผลของการทำนายการแข่งขัน คือให้แสดงคำว่า WIN เมื่อทีมแรกชนะ, คำว่า LOSE เมื่อทีมแรกแพ้ หรือ DRAW เมื่อผลออกมาเสมอกัน

Sample Input	Sample Output
3 Spain Netherlands 3 1 3 2 1 1 2 2 Germany Spain 3 1 1 4 0 1 Thai Brazil 1 1 2 2 5 0 0 11 6 6	WIN LOSE DRAW

<u>คำอธิบายเพิ่มเติม</u>

จากตัวอย่างมีผลที่ต้องทำนาย 3 คู่ คู่แรกเป็นการแข่งระหว่าง Spain และ Netherlands ผลการคำนวนคะแนนการทำนายได้ค่า +2 นั่นคือทีม Spain มีความเป็นไปได้อย่างมากที่จะชนะ จึงให้ผลลัพธ์เป็น WIN