TecnoMate

COMPETENCIA DE PROGRAMACIÓN

Nivel “Paradigmas” - Problem Set

14 de Octubre de 2016

Problem 1: Vonny and her dominoes

SPOJ code: VONNY

Vonny loves playing with dominoes. And so she owns a standard set of dominoes. A standard set of dominoes consists of 28 pieces called bones, tiles or stones. Each bone is a rectangular tile with a line dividing its face into two square ends. Each square is labeled with a number between 0 and 6. The 28 stones are labeled (0,0), (0,1), (0,2), (0,3), (0,4), (0,5), (0,6), (1,1), (1,2), ..., (5,5), (5,6), (6,6). Tommy - the brother of Vonny - build a box for Vonny's dominoes. This box is sized 7 x 8 squares. Every square is labeled with a number between 0 and 6. You can see a example box here.

```
0 3 0 2 2 0 2 3
1 5 6 5 5 1 2 2
3 4 1 4 5 4 4 4
6 6 1 0 5 2 3 0
4 0 3 2 4 1 6 0
1 4 1 5 6 6 3 0
1 2 6 5 5 6 3 3
```

Now Vonny wants to arrange her 28 stones in such way that her stones cover all squares of the box. A stone can only be placed on two adjacent squares if the numbers of the squares and of the domino stone are equal. Tommy asks Vonny in how many different ways she can arrange the dominos. Tommy assumes that Vonny need a lot of time to answer the question. And so he can take some of Vonny's candies while she solves the task. But Vonny is a smart and clever girl. She asks you to solve the task and keeps an eye on her candies.

Input

The first line of the input contains the number of testcases. Each case consists of 56 numbers (7 rows and 8 cols) between 0 and 6 which represents Tommy's box.

Output

For each testcase output a single line with the number which answers Tommy's question.

Example

Input	Output

2
0 3 0 2 2 0 2 3
1 5 6 5 5 1 2 2
3 4 1 4 5 4 4 4
6 6 1 0 5 2 3 0
4 0 3 2 4 1 6 0
1 4 1 5 6 6 3 0
1 2 6 5 5 6 3 3
 	
5 3 1 0 0 1 6 3
0 2 0 4 1 2 5 2
1 5 3 5 6 4 6 4
0 5 0 2 0 4 6 2
4 5 3 6 0 6 1 1
2 3 5 3 4 4 5 3
2 1 1 6 6 2 4 3
	18
	1

Problem 2: Printer Queue

SPOJ code: PQUEUE

The only printer in the computer science students' union is experiencing an extremely heavy workload. Sometimes there are a hundred jobs in the printer queue and you may have to wait for hours to get a single page of output.

Because some jobs are more important than others, the Hacker General has invented and implemented a simple priority system for the print job queue. Now, each job is assigned a priority between 1 and 9 (with 9 being the highest priority, and 1 being the lowest), and the printer operates as follows.

- The first job J in queue is taken from the queue.
- If there is some job in the queue with a higher priority than job J, then move J to the end of the queue without printing it.
- Otherwise, print job J (and do not put it back in the queue).

In this way, all those important muffin recipes that the Hacker General is printing get printed very quickly. Of course, those annoying term papers that others are printing may have to wait for quite some time to get printed, but that's life.

Your problem with the new policy is that it has become quite tricky to determine when your print job will actually be completed. You decide to write a program to figure this out. The program will be given the current queue (as a list of priorities) as well as the position of your job in the queue, and must then calculate how long it will take until your job is printed, assuming that no additional jobs will be added to the queue. To simplify matters, we assume that printing a job always takes exactly one minute, and that adding and removing jobs from the queue is instantaneous.

Input

One line with a positive integer: the number of test cases (at most 100). Then for each test case:

- One line with two integers n and m, where n is the number of jobs in the queue ($1 \leq n \leq 100$) and m is the position of your job ($0 \leq m \leq n-1$). The first position in the queue is number 0, the second is number 1, and so on.
- One line with n integers in the range 1 to 9, giving the priorities of the jobs in the queue. The first integer gives the priority of the first job, the second integer the priority of the second job, and so on.

Output

For each test case, print one line with a single integer; the number of minutes until your job is completely printed, assuming that no additional print jobs will arrive.

Example

Input	Output

3
1 0
5
4 2
1 2 3 4
6 0
1 1 9 1 1 1
	1
	2
	5

Problem 3: Scavenger Hunt

SPOJ code: SCAVHUNT

Bill has been the greatest boy scout in America and has become quite a superstar because he always organized the most wonderful scavenger hunts (you know, where the kids have to find a certain route following certain hints). Bill has retired now, but a nationwide election quickly found a successor for him, a guy called George. He does a poor job, though, and wants to learn from Bill's routes. Unfortunately Bill has left only a few notes for his successor. Bill never wrote his routes completely, he only left lots of little sheets on which he had written two consecutive steps of the routes. He then mixed these sheets and memorized his routes similarly to how some people learn for exams: practicing again and again, always reading the first step and trying to remember the following. This made much sense, since one step always required something from the previous step. George however would like to have a route written down as one long sequence of all the steps in the correct order. Please help him make the nation happy again by reconstructing the routes.

Input

The first line contains the number of scenarios. Each scenario describes one route and its first line tells you how many steps ($3 \leq S \leq 333$) the route has. The next $S - 1$ lines each contain one consecutive pair of the steps on the route separated by a single space. The name of each step is always a single string of letters.

Output

The output for every scenario begins with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print S lines containing the steps of the route in correct order. Terminate the output for the scenario with a blank line.

Example

Input	**Output**

```
2
4
SwimmingPool OldTree
BirdsNest Garage
Garage SwimmingPool
3
Toilet Hospital
VideoGame Toilet
```
	```
	Scenario #1:
	BirdsNest
	Garage
	SwimmingPool
	OldTree

	Scenario #2:
	VideoGame
	Toilet
	Hospital
	```

Problem 4: Termites Strike Back

SPOJ code: TERMITES

Princess Bala and Z founded a new colony where ants are free to express their own beliefs. The new colony, however, now faces a serious threat to its survival. Acid-shooting termites have become stronger and are willing to attack the nest.

Z needs to develop a defense strategy for the colony. Queen Ant, Princess Bala's mother, tells him that at least twelve ants are needed to successfully kill a single termite. Unfortunately, the number of soldiers has had a significant decrease since the new freedom ideals were spread out. Thus, he decides that no more ants than needed will participate in the battle.

In order to create a safety perimeter, soldiers need to be set in concentric rings around the colony's nest. The number of warriors of the innermost ring must be a multiple of four. Besides, to coordinate the defense each ring must have exactly 8 ants more than the ring it surrounds. Taking into account these constraints, the number of rings must be as large as possible.

Z's friend, Weaver, has been sent to the enemy's territory to spy them and gather vital information about their plans. Being successful, he will inform Z the exact number of warriors they are going to send. Given this information Ant Z needs all the help you can provide him to determine the best deployment for his troops.

Input

The input consists of several test cases, each one in a single line. Each test case consists of the number of termites n (1 <= n < 232) willing to attack the nest. The input terminates when n = 0.

Output

For each test case, indicate the number of soldiers needed on the outmost defense ring.

Example

Input	Output
1 2 33 0	12 16 76

Problem 5: Funny Plant

SPOJ code: FUNPLANT

Scientists have discovered a new plant. The fruit of the plant can feed 1 person for a whole week and, best of all, the plant never dies. Fruits need 1 week to grow, so each week you can harvest the fruits. Also the plant gives 1 fruit more than the week before and to get more plants you need to plant a fruit.

Now you need to calculate after how many weeks you can support a group of x people, given y fruits to start with.

Input description

The first line is a single positive integer t. The next t lines contain 2 positive integers x and y, being x the number of people needed to be fed and y the number of fruits you start with, where 1 <= x, y <= 7,400,000,000 (the approximate world's population).

Output description

For each pair of numbers x and y, indicate in a single line the number of weeks before you can feed the entire group of people.

Explanation

Here you have a table that shows the growth when starting with 1 fruit. It shows when the plant came into existence (is planted) and how many fruit it bears each week.

Plant Week	1	2	3	4	5	6	7	8	9	10	11	12	13	Total fruits for the week
1	0	-	-	-	-	-	-	-	-	-	-	-	-	0
2	1	0	-	-	-	-	-	-	-	-	-	-	-	1
3	2	1	0	0	0	-	-	-	-	-	-	-	-	3
4	3	2	1	1	1	0	0	0	0	0	0	0	0	8
5	4	3	2	2	2	1	1	1	1	1	1	1	1	21

At week 1 we have 1 plant giving 0 fruits, because it has just been planted.

When week 2 comes along we have 1 plant that gives off a fruit and then we use that fruit to plant plant 2.

Then in week 3 we have 2 fruits from plant 1, 1 from plant 2, so we can plant 3 new plants.

Example

Input	Output

3
200 15
50000 1
150000 250
	5
	14
	9

Problem 6: Switches

SPOJ code: SWITCHES

When you were a little kid, was indiscriminately flicking light switches super fun? I know it was for me. Let's tap into that and try to recall that feeling with today's challenge.

Imagine a row of N light switches, each attached to a light bulb. All the bulbs are off to start with. You are going to release your inner child so they can run back and forth along this row of light switches, flipping bunches of switches from on to off or vice versa. The challenge will be to figure out the state of the lights after this fun happens.

Input description

The first line is a single positive integer t, which indicates the number of test cases that follows.

Each test case will have two parts. First, the number of switches/bulbs N (1 <= N <= 1000) and the number of passes of your inner child M (1 <= M <= 100) are specified in a single line. On the remaining lines, there will be pairs of integers indicating ranges of switches that your inner child toggles as he runs back and forth. These ranges are inclusive (both endpoints, along with everything between them, are included), and the positions of switches are zero-indexed (so the possible positions range from 0 to N-1).

Output description

For each test case, the output will include a single line with a number: the number of switches that are on after all the running around.

Example

There is a more thorough explanation of what happens below.

Input	**Output**

1
10 4
3 6
0 4
7 3
9 9
	7

Explanation of example:

Below is a step by step rendition of which switches each range toggled in order to get the output described above.

```
    0123456789
    ..........
3-6    ||||
    ...XXXX...
0-4 |||||
    XXX..XX...
7-3    |||||
    XXXXX..X..
9-9          |
    XXXXX..X.X
```

As you can see, 7 of the 10 bulbs are on at the end.

Problem 7: Peer Review

SPOJ code: PEERVIEW

For scientific conferences, scientists submit papers presenting their ideas, and then review each other's papers to make sure only good papers are presented at the conference. Each paper must be reviewed by several scientists, and scientists must not review papers written by people they collaborate with (including themselves), or review the same paper more than once.

You have been asked to write a program to check if your favorite conference is doing things right. Whether a paper is being reviewed too much, too little, or by the wrong people - the organizers must know before it is too late!

Input

The first line in each test case has two integers, K (1 <= K <= 5) and N (1 <= N <= 1000). K is the number of reviews that each paper will receive, while N is the number of papers to be reviewed. The conference only accepts papers with a single author, and authors can only present a single paper at the conference.

Each of the next N lines describes an author and includes the name of the institution to which the author belongs, followed by the list of the K papers he or she has been requested to review. It is assumed that researchers from the same institution collaborate with each other, whereas researchers from different institutions don't. All institution names are shorter than 10 characters, and contain only upper or lowercase letters and no whitespace. Since we have as many papers as authors, papers are identified by their author's index; paper 1 was written by the first author in the list, and paper N was written by the last author.

The end of the test cases is marked with a line containing K = 0 and N = 0. You should generate no output for this line.

Output

For each test case, your program should output `NO PROBLEMS FOUND` (if all rules are being followed) or `P PROBLEMS FOUND`, where P is the number of rule violations found (counting at most 1 violation per paper). If there is exactly one rule violation overall, your program should output `1 PROBLEM FOUND`.

Example

Input	Output
2 3 UCM 2 3 UAM 1 3 UPM 1 2 2 3 UCM 2 3 UAM 1 2 UPM 2 2 0 0	NO PROBLEMS FOUND 3 PROBLEMS FOUND