PROGRAMMING LEAGUE 2016
FINAL
OPEN CATEGORY

Sponsored by,

Instructions:

1. The contest consists **6 questions**.
2. You are given **3 hours** to answer the questions.
3. You are **not** expected to answer all the questions.
4. You are **not allowed** to use any electronic devices throughout the contest except your calculator.
5. You are **allowed** to use physical references throughout the contest such as printed materials/ reference books.
6. Rankings will be based on the number of questions solved, number of attempts to solve a question and the time taken for a question to be solved.
7. Plan your strategy wisely.

Questions:

Question	Title	Contributor/Source
A	Family Day!	Khooi Xin Zhe
B	Hexa Addition	IIUM Code Knights
C	Go Dutch	Khooi Xin Zhe
D	Pair Sum	ACM ICPC MY 2010 (F)
E	Mini Sudoku X	ACM ICPC MY 2010 (E)
F	The Sultan's Chapati	ACM ICPC MY 2013 (E)

A	FAMILY DAY!	
	Input	Standard Input
	Output	Standard Output
	Time Limit	2 seconds

Problem Description

It's family day! Johnny and his son are watching The Bugs Bunny show cartoon on Cartoon Network.

"Woooooo, you scwewy wabbit!" said Daffy the Duck.

Johnny is teaching his son to count the number of words in the subtitile. Can you write a program to help him to display the word counts?

Input

Input to your program will consist of a series of lines, each line containing multiple words (at least one). A "word" is defined as a ***consecutive sequence of letters*** (upper and/or lower case).

Output

Your program should output a word count for each line of input. Each word count should be printed on a separate line.

Sample Input

Sample Input
Meep Meep I tot I taw a putty tat. I did! I did! I did taw a putty tat.

Sample Output

Sample Output
2 7 10

B	HEXA ADDITION	
	Input	Standard Input
	Output	Standard Output
	Time Limit	2 seconds

Problem Description

Johnny is a very hardworking student. He likes to add numbers in different numeral systems. He knows how to add two decimal numbers, for example, 22 + 5 = 27. Furthermore, he knows how to add two binary numbers, octal numbers. However, he stumbles upon the addition of two numbers in hexadecimal.

As a good friend of Johnny, he came to ask for your help! Can you help him? Please.

You are given two hexadecimal number, you are required to output their addition in hexadecimal.

Input

The first line on input consists of one positive integer. N, where N <= 100,000,000 which specifies the number of test cases.

For each test cases, there will be values ***x*** and ***y*** that you are required to do the summation in hex.

The values could be as large as 15 digits hexadecimal value and as low as 1 digit hexadecimal value. You can assume that the Input hexadecimal values are all in uppercase.

Output

For each test case, the output is in the format of "x + y = z", where ***x*** and ***y*** are the input values and ***z*** is the sum that you have calculated in hexadecimal.

Sample Input Output

Sample Input	Sample Output
2 3 3 F F	3 + 3 = 6 F + F = 1E

C	GO DUTCH	
	Input	Standard Input
	Output	Standard Output
	Time Limit	2 seconds

Problem Description

Johnny and his friends, Vincent, Elvin and Nick always eat out at premium restaurants.

Each of the time, Johnny will always pay for the bill first, then his friends will pay him later. However, it is always troublesome to divide the bill among themselves due to the annoying 6% GST and 10% Service Charge.

Your task is to write a program to help Johnny and his friends to calculate out the amount that each of them should pay.

Note: Final_Price = Total_Price * 1.16

Input

The input consists of a few test cases. For each test cases, the first line of input is a string which indicates the resturants name. The following line is a positive integer, n, ($1 \leq n \leq 10$) which indicates the number of lines in their receipt.

Each of the receipt lines are formatted according to the format below:

FOOD_CODE PRICE_TOTAL NUMBER_PEOPLE_SHARING NAME_SHARING

Example:

```
F001 40.00 2 Johnny Vincent
```

The program terminates when the input for the restaurant name is "#".

Output

For each test cases, the price calculated should be rounded off to the nearest whole number.

Output a line in the format "`RESTAURANT_NAME: Johnny RM`***`w`*** `Vincent RM`***`x`*** `Elvin RM`***`y`*** `Nick RM`***`z`***".

The price should be formatted in 2 decimal places, e.g. RM99.00.

Sample Input Output

Sample Output

```
SUSHI PRINCE
4
F001 40.00 2 Johnny Vincent
F002 19.90 1 Johnny
F003 9.90 2 Nick Elvin
D001 10.00 4 Johnny Vincent Elvin Nick
STEAK BAR
3
F001 120.00 4 Johnny Vincent Elvin Nick
D001 36.00 2 Johnny Nick
D002 40.00 2 Elvin Vincent
#
```

Sample Output

```
SUSHI PRINCE: Johnny RM49.00 Vincent RM26.00 Elvin RM9.00 Nick RM9.00
STEAK BAR: Johnny RM56.00 Vincent RM58.00 Elvin RM58.00 Nick RM56.00
```

D	PAIRED SUM	
	Input	Standard Input
	Output	Standard Output
	Time Limit	3 seconds

Problem Description

You are given an integer array of size N and an integer M. This array has (N*(N-1))/2 different pairs. You need to calculate how many of those pairs have the sum equal to M. For example, if the array is {1,2,3,4} and M is 5, then there are exactly 2 pairs {1,4} and {2,3} whose sum is equal to M.

Input

The first line has a positive integer T, T <= 100,000, denoting the number of test cases. This is followed by each test case per line. Each test case consists of two lines. The first line contains 2 integers N and M separated by a single space. N is between 2 and 20,000 inclusive. The second line consists of the N integers separated by a single space, denoting the values of the given array. All the numbers in the array are between 1 and 1,000,000,000 inclusive. They are distinct and sorted in increasing order.

Output

For each test case, the output contains a line in the format Case #x: R, where x is the case number (starting from 1) and R is the number of pairs whose sum is exactly the given M.

Sample Input Output

Sample Input	Sample Output
5 8 100 19 25 32 48 52 68 75 81 8 100 19 28 31 49 51 61 72 81 8 100 16 22 38 46 58 62 73 81 8 100 13 21 32 48 52 67 78 87 8 100 13 24 34 43 57 61 76 81	Case #1: 4 Case #2: 3 Case #3: 1 Case #4: 2 Case #5: 2

E	MINI SUDOKU X	
	Input	Standard Input
	Output	Standard Output
	Time Limit	5 seconds

Problem Description

In Mini Sudoku X, there are 6 x 6 boxes to be filled with digits so that each row, column, main diagonal, and 2 x 3 square contains all the digits from 1 to 6. An example of a solution is as follows with the main diagonals shaded:

6	5	1	4	3	2
2	3	4	1	5	6
4	2	5	3	6	1
3	1	6	2	4	5
5	4	2	6	1	3
1	6	3	5	2	4

Write a program that reads a series of 36 digits representing a solution for Mini Sudoku X, and determines whether the solution is correct.

Input

The first line has a positive integer T, T <= 100000, denoting the number of test cases. This is followed by each test case per line. Each test case consists of six lines. Each line of the test case contains six 1-digit integers separated by a space.

Output

For each test case, the output contains a line in the format Case #x: M, where x is the case number (starting from 1) and M is either 0 or 1. 1 if the test case represents a correct solution and 0 otherwise.

Sample Input Output

Sample Input	Sample Output
10 1 1 4 3 6 4 3 2 3 5 2 1 4 2 3 2 2 3 2 6 4 3 3 2 2 3 1 2 3 3 4 6 3 1 3 5 2 5 3 3 5 2 2 1 6 4 4 4 5 6 1 1 5 2 1 4 3 4 1 1 6 1 5 3 4 3 4 1 3 2 5 1 2 6 4 3 4 4 1 2 2 4 5 3 1 3 3 2 6 1 2 6 3 2 2 4 4 3 5 1 1 2 4 3 6 2 5 5 4 1 6 2 4 3 3 1 1 1 1 1 1 2 1 1 6 1 4 6 3 5 1 5 5 3 2 2 1 1 4 6 4 1 2 3 2 1 6 1 1 2 3 5 4 2 3 4 3 2 4 2 5 3 3 3 2 4 4 6 4 4 4 3 4 3 4 4 3 1 4 3 3 4 1 6 2 5 6 5 2 4 3 1 1 3 4 2 5 6 5 2 6 1 4 3 2 1 3 5 6 4 4 6 5 3 1 2 3 5 4 2 3 1 5 2 1 3 2 4 4 1 3 6 2 6 2 3 4 1 2 4 3 2 4 4 3 1 5 2 5 4 2 4 3 2 1 3 2 5 2 4 4 4 2 2 1 6 4 5 1 5 6 6 3 3 6 3 1 4 4 6 3 2 3 3 1 2 4 1 1 3 4 4 3 2 3 6 1 4 1 1 3 4 5 1 2 5 2 1 5 1 3 2 2 6 3 1 2 2	Case#1: 0 Case#2: 0 Case#3: 0 Case#4: 0 Case#5: 0 Case#6: 1 Case#7: 0 Case#8: 0 Case#9: 0 Case#10: 0

4 2 2 2 2 1 4 1 3 5 3 4 1 5 4 4 3 3 4 5 1 2 5 1 4 1 5 5 1 2 3 5 3 6 2 6 3 4 6 3 3 5	

F	THE SULTAN'S CHAPATI	
	Input	Standard Input
	Output	Standard Output
	Time Limit	3 seconds

Problem Description

The Sultan of Isketambola likes his Chapati served to him in a stack of uniquely sized Chapatis and heated up in a special way. The cook for the Sultan would have to heat up a stack of uniquely sized Chapati where each Chapati on the bottom of the stack is larger in diameter to the one above.

You are to write a program that indicates how the stack can be sorted so that the largest Chapati is on the bottom and the smallest Chapati is on the top. The size of a Chapati is given by the Chapati's diameter. All Chapatis in a stack have different diameters.

Sorting a stack is done by a sequence of Chapati "flips". A flip consists of inserting a spatula between two Chapatis in a stack and flipping (reversing) the Chapatis on the spatula (reversing the sub-stack). A flip is specified by giving the position of the Chapati on the bottom of the sub-stack to be flipped (relative to the whole stack). The Chapati on the bottom of the whole stack has position 1 and the

Chapati on the top of a stack of *n* Chapatis has position *n*.

A stack is specified by giving the diameter of each Chapati in the stack in the order in which the Chapatis appear. For example, consider the three stacks of Chapatis below (in which Chapati 8 is the top-most Chapati of the left stack):

8 2 4

6 4 2

1 1 1

4 6 6

2 8 8

The stack on the left can be transformed to the stack in the middle via *flip(1)*. The middle stack can be transformed into the right stack via the command *flip(4)*. A final flip of flip(3) will result in a sorted stack.

Input

First line of the input contains T the number of test cases (1<=T<=1000). Each test case consists two lines. First line of the test case contains N (1<=N<=30), the number of Chapatis in the stack. The next line contains N integers separated by a single space. Each integer specifies the diameter of the Chapati between 1 and 100 from the top most position to the bottom.

Output

For each test case, the output contains a line in the format Case #x: followed by a sequence of integers, where x is the case number (starting from 1). For each stack the sequence of flips should be terminated by a 0 (indicating no more flips are necessary). Once a stack is sorted, no more flips should be made.

Sample Input Output

Sample Input	Sample Output
3 5 1 2 3 4 5 5 5 4 3 2 1 5 8 6 1 4 2	Case #1: 0 Case #2: 1 0 Case #3: 1 4 3 0