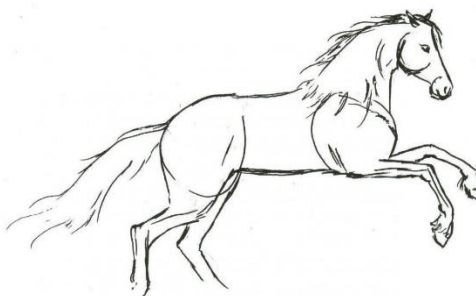


Problem A: Horse race (blue balloon)



Mamane regularly organizes horse race. But he needs your help to test the effectiveness of his chronometer.

Given the length **L** of the race circuit into meter and average speed of the horse **V** into meters per second (*m/s*), your task is to write a program that will determine the time it takes the horse in hours, minutes and seconds, in format **HH:MM:SS** where **HH** is a two-digit number representing the number of hours ; **MM** ($MM < 60$) , a two-digit number representing the number of minutes, and **SS** ($SS < 60$), a two-digit number representing the number of seconds. SS and MM must each have the greatest possible value. In this problem, L and V are integers and HH will be strictly lower than 24.

Standard Input

The first line of input contains a single integer **P**, ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set consists of one line containing the length L, the space and the average speed V of the horse.

Standard Output

For each data set, generate one line of output containing the time in format **HH:MM:SS**.

Sample Input	Sample Output
4	00:00:40
1000 25	00:00:00
20 30	00:03:01
10000 55	05:25:10
390200 20	

Problem B: Seconds (red balloon)



Your task is to convert the time in format **HH:MM:SS**, in seconds.

- **HH** is a two-digit number representing the number of hours, which is lower than 24
- **MM** ($MM < 60$), a two-digit number representing the number of minutes
- **SS** ($SS < 60$), a two-digit number representing the number of seconds

For example 00:00:40 equals to 40 seconds and 05:25:10 equals to 19510 seconds.

Standard Input

The first line of input contains a single integer **P**, ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set consists of one line containing the time in format **HH:MM:SS**.

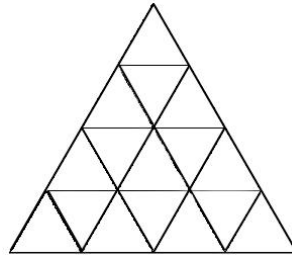
Standard Output

For each data set, generate one line of output in the form of “**HH:MM:SS** equals to n seconds” where n is the time in seconds.

Sample Input	Sample Output
3 00:00:40 05:25:10 09:18:18	00:00:40 equals to 40 seconds 05:25:10 equals to 19510 seconds 09:18:18 equals to 33498 seconds

Problem C: Sub Triangles (yellow balloon)

Find the number of triangles of length **1**, contained in an equilateral grid triangle of length l . A grid triangle of length 4 will look like this:



The total number of triangles of length **1** which can be seen in this image is 16. Your task is to find the total number of triangles of length **1** which can be seen in an image of an equilateral grid triangle of length l .

Standard Input

The input will begin with a single integer **P** on the first line, indicating the number of cases that will follow.

The remaining lines of the input will consist of one integer l per line, which is the equilateral grid triangle length. All integers will be less than 1,000,000 and greater than 0. The length l will be less than 100,000.

You should process all integers and for each integer l , determine the total number of triangles of length **1** which can be seen in an image of an equilateral grid triangle of length l .

You can assume that no operation overflows a 32-bit integer.

Standard Output

For each integer l , you should output the total number of triangles of length **1** which can be seen in an image of an equilateral grid triangle of length l , with one line of output for each line of input.

Sample Input	Sample Output
4	1
1	4
2	9
3	16
4	

Problem D: Sums (green balloon)

The n^{th} *Triangular number*, $T(n) = 1 + \dots + n$, is the sum of the first n integers. It is the number of points in a triangular array with n points on side. For example $T(4)$:

```

      x
    x x
  x x x
x x x x
  
```

Write a program to compute the weighted sum of triangular numbers:

$$W(n) = \text{SUM}[k = 1..n; k * T(k+1)]$$

Standard Input

The first line of input contains a single integer N , ($1 \leq N \leq 1000$) which is the number of datasets that follow.

Each dataset consists of a single line of input containing a single integer n , ($1 \leq n \leq 300$), which is the number of points on a side of the triangle.

Standard Output

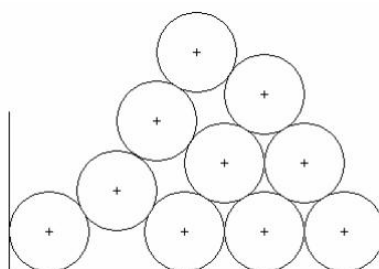
For each dataset, output on a single line the dataset number, (1 through N), a blank, the value of n for the dataset, a blank, and the weighted sum, $W(n)$, of triangular numbers for n .

Sample Input	Sample Output
4	1 3 45
3	2 4 105
4	3 5 210
5	4 10 2145
10	

Source: GreaterNY2006

Problem E : Cylinders (white balloon)

Cylinders (e.g. oil drums) (of radius 1 foot) are stacked in a rectangular bin. Each cylinder on an upper row rests on two cylinders in the row below. The cylinders in the bottom row rest on the floor and do not roll from their original positions. Each row has one less cylinder than the row below.



This problem is to write a program to compute the location of the center of the top cylinder from the centers of the cylinders on the bottom row. Computations of intermediate values should use double precision.

Standard Input


The input begins with a line containing the count of problem instances, ***nProb***, as a decimal integer, ($1 \leq nProb \leq 1000$). This is followed by ***nProb*** input lines. An input line consists of the number, ***n***, of cylinders on the bottom row followed by ***n*** floating point values giving the ***x*** coordinates of the centers of the cylinders (the ***y*** coordinates are all 1.0 since the cylinders are resting on the floor ($y = 0.0$)). The value of ***n*** will be between 1 and 10 (inclusive). The distance between adjacent centers will be at least **2.0** (so the cylinders do not overlap) and at most **3.4** (so cylinders at level ***k*** cannot touch cylinders at level ***k*** - 2).

Standard Output

The output for each data set is a line containing the problem number (1...***nProb***), a colon, a space, the ***x*** coordinate of the topmost cylinder to 4 decimal places, a space and the ***y*** coordinate of the topmost cylinder to 4 decimal places. **Note:** To help you check your work, the ***x***-coordinate of the center of the top cylinder should be the average of the ***x***-coordinates of the leftmost and rightmost bottom cylinders.

Sample Input	Sample Output
5	1: 6.1000 4.1607
4 1.0 4.4 7.8 11.2	2: 1.0000 1.0000
1 1.0	3: 6.0000 9.6603
6 1.0 3.0 5.0 7.0 9.0 11.0	4: 10.7000 15.9100
10 1.0 3.0 5.0 7.0 9.0 11.0 13.0 15.0 17.0 20.4	5: 7.8000 5.2143
5 1.0 4.4 7.8 11.2 14.6	

Source: GreaterNY2005

	<p>The 2016 Nigerian Collegiate Programming Contest</p>	<p>UAM 9th April 2016 (C, C++, JAVA)</p>
--	---	--

Problem F : Base Conversion

(orange balloon)

Write a program to convert numbers in one base to numbers in a second base. There are 62 different digits: { 0-9,A-Z,a-z }

HINT: If you make a sequence of base conversions using the output of one conversion as the input to the next, when you get back to the original base, you should get the original number.

Standard Input

The first line of input contains a single positive integer. This is the number of lines that follow. Each of the following lines will have a (decimal) input base followed by a (decimal) output base followed by a number expressed in the input base. Both the input base and the output base will be in the range from 2 to 62. That is (in decimal) **A = 10, B = 11, ..., Z = 35, a = 36, b = 37, ..., z = 61** (0-9 have their usual meanings).

Standard Output


The output of the program should consist of three lines of output for each base conversion performed. The first line should be the input base in decimal followed by a space then the input number (as given expressed in the input base). The second output line should be the output base followed by a space then the input number (as expressed in the output base). The third output line is blank.

Sample Input

```

8
62 2 abcdefghiz
10 16 1234567890123456789012345678901234567890
16 35 3A0C92075C0DBF3B8ACBC5F96CE3F0AD2
35 23 333YMHOUE8JPLT7OX6K9FYCQ8A
23 49 946B9AA02MI37E3D3MMJ4G7BL2F05
49 61 1VbDkSIMJL3JjRgAdlUfcaWj
61 5 dl9MDSWqwhjDnToKcsWE1S
5 10 42104444441001414401221302402201233340311104212022133030

```

	<p><i>The 2016 Nigerian Collegiate Programming Contest</i></p>	<p><i>UAM 9th April 2016 (C, C++, JAVA)</i></p>
--	--	---

Sample Output

```

62 abcdefghiz
2 1011100000100010111110010010110011111001001100011010010001

10 1234567890123456789012345678901234567890
16 3A0C92075C0DBF3B8ACBC5F96CE3F0AD2

16 3A0C92075C0DBF3B8ACBC5F96CE3F0AD2
35 333YMHOUE8JPLT7OX6K9FYCQ8A

35 333YMHOUE8JPLT7OX6K9FYCQ8A
23 946B9AA02MI37E3D3MMJ4G7BL2F05

23 946B9AA02MI37E3D3MMJ4G7BL2F05
49 1VbDkSIMJL3JjRgAdlUfcaWj

49 1VbDkSIMJL3JjRgAdlUfcaWj
61 dl9MDSWqwHjDnToKcsWE1S

61 dl9MDSWqwHjDnToKcsWE1S
5 42104444441001414401221302402201233340311104212022133030

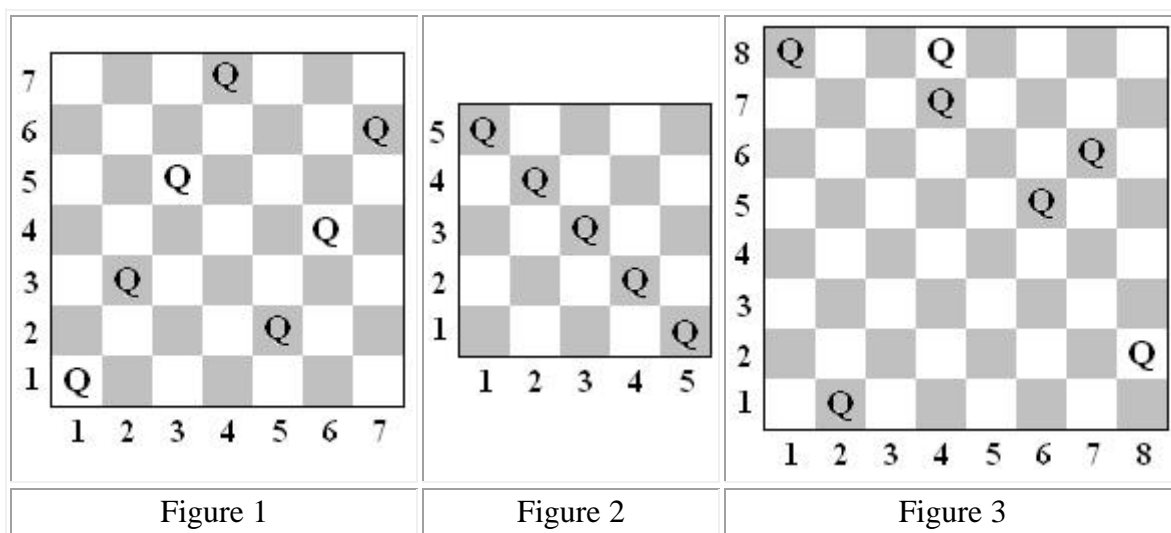
5 42104444441001414401221302402201233340311104212022133030
10 1234567890123456789012345678901234567890

```

Source: GreaterNY2002

Problem G: Collisions

(black balloon)



Lots of time has been spent by computer science students dealing with queens on a chess board. Two queens on a chessboard *collide* if they lie on the same row, column or diagonal, and there is no piece between them. Various sized square boards and numbers of queens are considered. For example, Figure 1, with a 7 x 7 board, contains 7 queens with no collisions. In Figure 2 there is a 5 x 5 board with 5 queens and 4 collisions. In Figure 3, a traditional 8 x 8 board, there are 7 queens and 5 collisions.

On an $n \times n$ board, queen positions are given in Cartesian coordinates (x, y) where x is a column number, 1 to n , and y is a row number, 1 to n . Queens at distinct positions (x_1, y_1) and (x_2, y_2) lie on the same diagonal if $(x_1 - x_2)$ and $(y_1 - y_2)$ have the same magnitude. They lie on the same row or column if $x_1 = x_2$ or $y_1 = y_2$, respectively. In each of these cases the queens have a collision if there is no other queen directly between them on the same diagonal, row, or column, respectively. For example, in Figure 2, the collisions are between the queens at (5, 1) and (4, 2), (4, 2) and (3, 3), (3, 3) and (2, 4), and finally (2, 4) and (1, 5). In Figure 3, the collisions are between the queens at (1, 8) and (4, 8), (4, 8) and (4, 7), (4, 7) and (6, 5), (7, 6) and (6, 5), and finally (6, 5) and (2, 1). Your task is to count queen collisions.

In many situations there are a number of queens in a regular pattern. For instance in Figure 1 there are 4 queens in a line at (1,1), (2, 3), (3, 5), and (4, 7). Each of these queens after the first at (1, 1) is one to the right and 2 up from the previous one. Three queens starting at (5, 2) follow a similar pattern. Noting these patterns can allow the positions of a large number of queens to be stated succinctly.

Standard Input

The input will consist of one to twenty data sets, followed by a line containing only 0.

The first line of a dataset contains blank separated positive integers n g , where n indicates an $n \times n$ board size, and g is the number of linear patterns of queens to be described, where $n < 30000$, and $g < 250$. The next g lines each contain five blank separated integers, k x y s t , representing a linear pattern of k queens at locations $(x + i*s, y + i*t)$, for $i = 0, 1, \dots, k-1$. The value of k is positive. If k is 1, then the values of s and t are irrelevant, and they will be given as 0. All queen positions will be on the board. The total number of queen positions among all the linear patterns will be no more than n , and all these queen positions will be distinct.

Standard Output

There is one line of output for each data set, containing only the number of collisions between the queens.

The sample input data set corresponds to the configuration in the Figures.

Take some care with your algorithm, or else your solution may take too long.

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

Source: MCPC 2010

	<i>The 2016 Nigerien Collegiate Programming Contest</i>	<i>UAM 9th April 2016 (C, C++, JAVA)</i>
--	---	--