



A • Sixth Grade Math

In sixth grade, students are presented with different ways to calculate the *Least Common Multiple (LCM)* and the *Greatest Common Factor (GCF)* of two integers. The **LCM** of two integers **a** and **b** is the smallest positive integer that is a multiple of both **a** and **b**. The **GCF** of two non-zero integers **a** and **b** is the largest positive integer that divides both **a** and **b** without remainder.

For this problem you will write a program that determines both the **LCM** and **GCF** for positive integers.

Input

The first line of input contains a single integer **N**, ($1 \leq N \leq 1000$) which is the number of data sets that follow. Each data set consists of a single line of input containing two positive integers, **a** and **b**, ($1 \leq a, b \leq 1000$) separated by a space.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), a space, the **LCM**, a space, and the **GCF**.

Sample Input	Sample Output
3	1 10 5
5 10	2 161 1
7 23	3 168 14
42 56	



B • Cryptoquote

A *cryptoquote* is a simple encoded message where one letter is simply replaced by another throughout the message. For example:

Encoded: **HPC PJVYMIY**
Decoded: **ACM CONTEST**

In the example above, **H=A**, **P=C**, **C=M**, **J=O**, **V=N**, **Y=T**, **M=E** and **I=S**. For this problem, you will decode messages.

Input

The first line of input contains a single integer N , ($1 \leq N \leq 1000$) which is the number of data sets that follow. Each data set consists of two lines of input. The first line is the encoded message. The second line is a 26 character string of upper case letters giving the character mapping for each letter of the alphabet: the first character gives the mapping for **A**, the second for **B** and so on. Only upper case letters will be used. Spaces may appear in the encoded message, and should be preserved in the output string.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), a space and the decoded message.

Sample Input	Sample Output
2 HPC PJVYMIY BLMRGJIASOPZEFDCCKWYHUNXQTV FDY GAI BG UKMY KIMHOTSQYRLCUZPAGWJNBVDXEF	1 ACM CONTEST 2 THE SKY IS BLUE



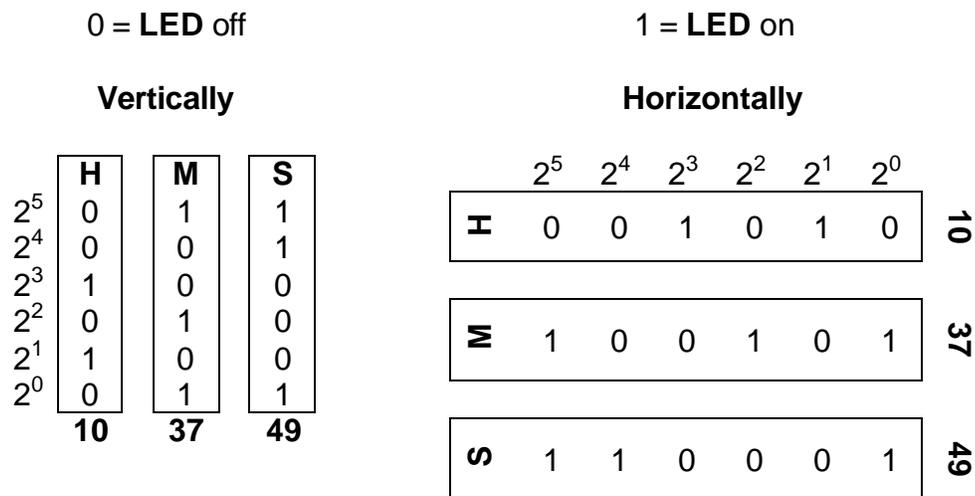
C • Binary Clock

A *binary clock* is a clock which displays traditional sexagesimal time (military format) in a binary format. The most common binary clock uses three columns or three rows of LEDs to represent zeros and ones. Each column (or row) represents a time-unit value.

When three columns are used (vertically), the bottom row in each column represents **1** (or 2^0), with each row above representing higher powers of two, up to 2^5 (or **32**). To read each individual unit (hours, minutes or seconds) in the time, the user adds the values that each illuminated LED represents, and then reads the time from left to right. The first column represents the hour, the next column represents the minute, and the last column represents the second.

When three rows are used (horizontally), the right column in each row represents **1** (or 2^0), with each column left representing higher powers of two, up to 2^5 (or **32**). To read each individual unit (hours, minutes or seconds) in the time, the user adds the values that each illuminated LED represents, and then reads the time from top to bottom. The top row represents the hour, the next row represents the minute, and the bottom row represents the second.

For example:



Time is: **10 : 37 : 49**

For this problem you will read a time in sexagesimal time format, and output both the vertical and horizontal binary clock values. The output will be formed by concatenating together the bits in each column (or row) to form two 18 character strings of **1**'s and **0**'s as shown below.



Greater New York
Programming Contest
St. Joseph's College
Patchogue, NY



10:37:49 would be written vertically as **011001100010100011** and horizontally as **001010100101110001**.

Input

The first line of input contains a single integer N , ($1 \leq N \leq 1000$) which is the number of data sets that follow. Each data set consists of a single line of input containing the time in sexagesimal format.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), a space, the binary time in vertical format (18 binary digits), a space and the binary time in horizontal format (18 binary digits).

Sample Input	Sample Output
2	1 011001100010100011 001010100101110001
10:37:49	2 0000000000000000001 000000000000000001
00:00:01	



D • Recursively Palindromic Partitions

A *partition* of a positive integer N is a sequence of integers which sum to N , usually written with plus signs between the numbers of the partition. For example

$$15 = 1+2+3+4+5 = 1+2+1+7+1+2+1$$

A partition is *palindromic* if it reads the same forward and backward. The first partition in the example is *not* palindromic while the second is. If a partition containing m integers is palindromic, its left half is the first $\lfloor m/2 \rfloor$ integers and its right half is the last $\lfloor m/2 \rfloor$ integers (which must be the reverse of the left half. ($\lfloor x \rfloor$ is the greatest integer less than or equal to x .)

A partition is *recursively palindromic* if it is palindromic and its left half is recursively palindromic or empty. Note that every integer has at least two recursively palindromic partitions one consisting of all ones and a second consisting of the integer itself. The second example above is also recursively palindromic.

For example, the recursively palindromic partitions of 7 are:

$$7, 1+5+1, 2+3+2, 1+1+3+1+1, 3+1+3, 1+1+1+1+1+1$$

Write a program which takes as input an integer N and outputs the number of recursively palindromic partitions of N .

Input

The first line of input contains a single integer N , ($1 \leq N \leq 1000$) which is the number of data sets that follow. Each data set consists of a single line of input containing a single positive integer for which the number of recursively palindromic partitions is to be found.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), a space and the number of recursively palindromic partitions of the input value.

Sample Input	Sample Output
3	1 4
4	2 6
7	3 60
20	



E • Text Messaging Improvement?

On a standard mobile phone the letters are distributed across the keys 2 through 9 as:

	2 ABC	3 DEF
4 GHI	5 JKL	6 MNO
7 PQRS	8 TUV	9 WXYZ

To enter the letter **C**, you press key 2 three times (seeing **A-B-C**). The number of keystrokes to enter a letter depends on where it is in the list of letters on its key.

The *Flathead Telephone Company* (FTC) is considering rearranging the letters on the keys to reduce the average number of keystrokes required to enter names etc. or send text messages. The letters must still appear in alphabetical order on the keys but different numbers of letters may appear on each key and possibly more keys could be used. FTC has several databases of letter frequencies used in different applications. For instance, it might help to move **S** from the **7** key to the **8** key. They need a program which is given the frequencies of the letters and a number of keys and returns the assignment of letters to keys with the smallest average number of keystrokes using the given frequencies. Each key used must have at least one letter and at most eight letters.

Input

The first line of input contains a single integer **N**, ($1 \leq N \leq 1000$) which is the number of data sets that follow. Each data set consists of three lines of input. The first line contains a single integer **K**, ($4 \leq K \leq 26$), the number of keys which are to be used. The second and third lines contain 13 decimal values each giving the percent frequency of the letters **A** through **Z** in order.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), the best average number of keystrokes to three decimal places, a space and the letters **A** through **Z**, for the best arrangement, in order with a single



Greater New York
Programming Contest
St. Joseph's College
Patchogue, NY

event sponsors



space at the break between letters on different keys. It is possible that the same input data set may produce different output.

Sample Input	
2	
8	
8.167	1.492 2.782 4.253 12.702 2.228 2.015 6.094 6.966 0.153 0.772 4.025 2.406
6.749	7.507 1.929 0.095 5.987 6.327 9.056 2.758 0.978 2.360 0.150 1.974 0.075
9	
1.0	10.0 11.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
10.0	10.0 10.0 10.0 10.0 10.0 11.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
Sample Output	
1	1.647 AB CD EFG HIJK LM NOPQ RS TUVWXYZ
2	1.570 A B CDEFG HIJKLM N OP QR STUV WXYZ



F • Extended Normal Order Sort

When sorted in standard order, strings with digits in them may not sort to where they are expected. For instance, **xyz100** precedes **xyz2**. In some applications such as listing files, *normal order sort* may be used where any string of digits in a character string is treated as a single digit with numerical value given by the digit string. For example, the following are in normal order:

XYZ001, XYZ2, XYZ003, XYZ08, XYZ23, XYZ100, XYZQ

We wish to extend normal order sort in two ways:

1. Lower case and upper case letters sort the same (with the upper case value).
2. If a plus (+) or minus (-) sign precedes a digit and does not follow a digit, it is considered part of the following number for sorting purposes.

So **123+456+7890** are three numbers separated by plus signs but **A+003** is the same as **A3**.

To do our sort, we will use a library sort routine but we need to furnish a comparison routine. Write a comparison routine which takes as input two strings of printable, non-space *ASCII* characters (`chr(33) - chr(126)`) and returns:

- 1 if the first string should precede the second in extended normal order
- 0 if the two strings are the same in extended normal order, or
- 1 if the first string should follow the second in extended normal order.

Input

The first line of input contains a single integer **N**, ($1 \leq N \leq 1000$) which is the number of data sets that follow. Each data set consists of a single line of input containing the two strings to be compared separated by a space.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), a space and **-1**, **0** or **1** depending on whether the first string precedes, is the same as, or follows the second string in extended normal order.



Greater New York
Programming Contest
St. Joseph's College
Patchogue, NY

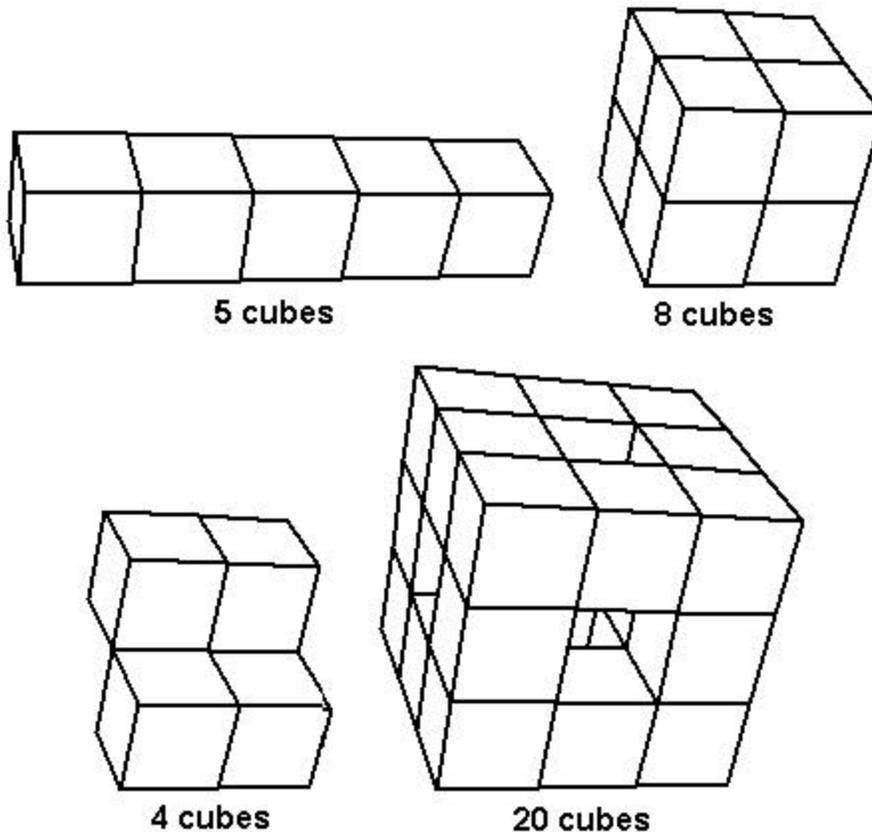
event sponsors



Sample Input	Sample Output
5	1 -1
x-3 X0001	2 1
123-456-7890 123+456+7890	3 0
xYz000123J XyZ+123j	4 -1
#\$%^&*[]- abcdefgh	5 0
Abc47jKL+00123 ABC+47jKL123	

G • Area of Polycubes

A *polycube* is a solid made by gluing together unit cubes (one unit on each edge) on one or more faces. The figure in the lower-left is *not* a *polycube* because some cubes are attached along an edge.



For this problem, the *polycube* will be formed from unit cubes centered at integer lattice points in 3-space. The *polycube* will be built up one cube at a time, starting with a cube centered at $(0,0,0)$. At each step of the process (after the first cube), the next cube must have a face in common with a cube previously included and not be the same as a block previously included. For example, a 1-by-1-by-5 block (as shown above in the upper-left *polycube*) could be built up as:

$(0,0,0)$ $(0,0,1)$ $(0,0,2)$ $(0,0,3)$ $(0,0,4)$

and a 2-by-2-by-2 cube (upper-right figure) could be built as:

$(0,0,0)$ $(0,0,1)$ $(0,1,1)$ $(0,1,0)$ $(1,0,0)$ $(1,0,1)$ $(1,1,1)$ $(1,1,0)$



Since the surface of the *polycube* is made up of unit squares, its area is an integer.

Write a program which takes as input a sequence of integer lattice points in 3-space and determines whether it correctly forms a *polycube* and, if so, what the surface area of the *polycube* is.

Input

The first line of input contains a single integer N , ($1 \leq N \leq 1000$) which is the number of data sets that follow. Each data set consists of multiple lines of input. The first line contains the number of points P , ($1 \leq P \leq 100$) in the problem instance. Each succeeding line contains the centers of the cubes, eight to a line (except possibly for the last line). Each center is given as 3 integers, separated by commas. The points are separated by a single space.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), a space and the surface area of the *polycube* if it is correctly formed, OR, if it is not correctly formed, the string "NO" a space and the index (starting with 1) of the *first* cube which does not share a face with a previous cube. Note that the surface area includes the area of any included holes.

Sample Input	Sample Output
4	1 22
5	2 24
0,0,0 0,0,1 0,0,2 0,0,3 0,0,4	3 NO 3
8	4 72
0,0,0 0,0,1 0,1,0 0,1,1 1,0,0 1,0,1 1,1,0 1,1,1	
4	
0,0,0 0,0,1 1,1,0 1,1,1	
20	
0,0,0 0,0,1 0,0,2 0,1,2 0,2,2 0,2,1 0,2,0 0,1,0	
1,0,0 2,0,0 1,0,2 2,0,2 1,2,2 2,2,2 1,2,0 2,2,0	
2,1,0 2,1,2 2,0,1 2,2,1	



H · The Two Note Rag

Since most computers are binary machines, both powers of two and problems that involve only two values are important to computer scientists. The following problem has to do with powers of two and the digits 1 and 2.

Some powers of two as decimal values, such as $2^9 = 512$ and $2^{89} = 618,970,019,642,690,137,449,562,112$ end in a string of digits consisting only of 1's and 2's (12 for 2^9 and 2112 for 2^{89}). In fact, it can be proved that:

*For every integer R , there exists a power of 2 such that 2^K uses **only** the digits 1 and 2 in its last R digits.*

This is shown a bit more clearly in the following table:

R	Smallest K	2^K
1	1	2
2	9	512
3	89	...112
4	89	...2112
5	589	...22112
6	3089	...122112
...

Your job is to write a program that will determine, for given R , the *smallest* K such that 2^K ends in a string of R digits containing only 1's and 2's.

Input

The first line of the input contains a single decimal integer, N , $1 \leq N \leq 50$, the number of problem data sets to follow. Each data set consists of a single integer R , $1 \leq R \leq 20$, for which we want a power of 2 ending in a string of R 1's and 2's.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), a space, the input value R , another space, and the smallest value K for which 2^K ends in a string of R 1's and 2's.



Greater New York
Programming Contest
St. Joseph's College
Patchogue, NY

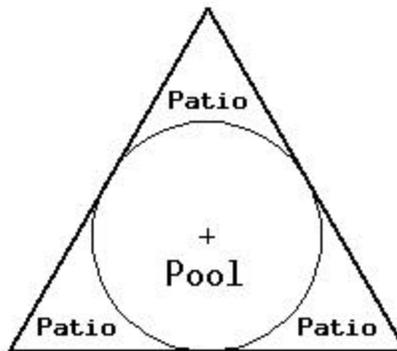
event sponsors



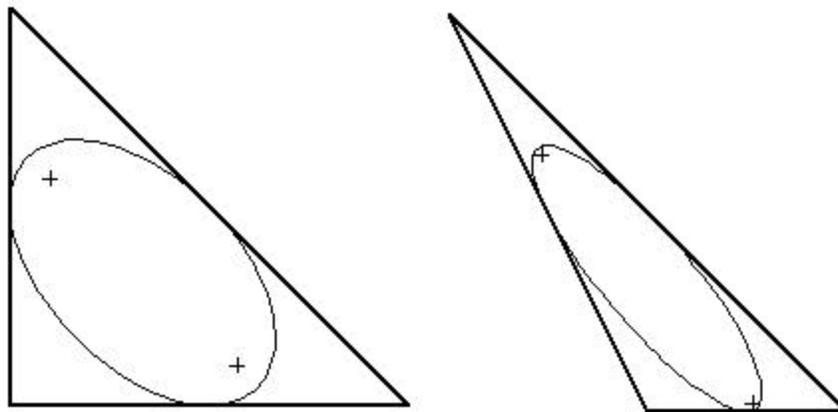
Sample Input	Sample Output
6	1 1 1
1	2 2 9
2	3 4 89
4	4 5 589
5	5 7 3089
7	6 15 11687815589
15	

I • Joe's Triangular Gardens

Joe's landscaping company specializes in gardens for computer geeks who have just had their company go public. One of his signature features is a round pool surrounded by a tiled patio in the form of an equilateral triangle where the edge of the pool is tangent to each side of the triangle at its midpoint.



Unfortunately, some of Joe's customers are not satisfied with an equilateral triangle, usually in the center of the garden. Some want it in a corner or next to a slope or some other layout. Joe would like the option of offering arbitrary triangular patios with an elliptical pool which is tangent to each side at the center of the side. For example:



Joe knows how to draw an ellipse by putting two stakes in the ground (at the foci of the ellipse), tying a rope between them and dragging a marker stick inside the rope. What Joe would like is for the customer to determine where the corners of the triangle will be and then measure the location of the triangle vertices and compute where to put the stakes and how long to make the rope.

Write a program, which takes as input the three vertices (x_1, y_1) , (x_2, y_2) and (x_3, y_3) of a triangle and computes an ellipse inscribed in the triangle, which is tangent to each side of the triangle at its



midpoint. The output is the coordinates of the two foci of the ellipse and the length of the rope (which is the sum of the distances from the foci to any point on the ellipse).

Input

The first line of input contains a single integer N , ($1 \leq N \leq 1000$) which is the number of data sets that follow. Each data set consists of a single line of input containing 6 space separated floating point numbers x_1 y_1 x_2 y_2 x_3 y_3 giving the coordinates of the vertices of a triangle.

Output

For each data set, you should generate one line of output with the following values: The data set number as a decimal integer (start counting at one), a space and five floating point values accurate to two decimal places each separated by a single space. The values are fx_1 fy_1 fx_2 fy_2 rl where (fx_1, fy_1) is one focus of the ellipse, (fx_2, fy_2) is the other focus of the ellipse and rl is the sum of the distances from the foci to any point on the ellipse (e.g. the length of the rope). The foci should be listed in *increasing lexicographical order* (i.e. $fx_1 \leq fx_2$ and if $fx_1 = fx_2$, $fy_1 \leq fy_2$). Note that in the case the ellipse is a circle, the two foci are the same (e.g. the center of the circle).

Sample Input	Sample Output
3	1 200.00 157.71 200.00 157.76 115.47
100 100 200 273.2051 300 100	2 119.53 213.81 213.81 119.53 163.30
100 100 100 300 300 100	3 103.94 253.14 229.40 146.86 170.51
100 200 100 300 300 100	