Problem A
Factors

(Time Limit: 3 seconds)

For a given number *n*, we would like to find the sum of all factors of *n*, including *n* itself. For example, the factors of the number **72** are: 1, 2, 3, 4, 6, 8, 9, 12, 18, 24, 36, 72. If you add them up, you will get the total of **195**. Please write a program to solve this problem.

Input Format

The first line is an integer indicating the number (≤ 10) of cases. For each case, the input *n* is an integer.

Output Format

For each test case, the output *sum* is also an integer. It means the sum of all factors of *n*.

Technical Specification

- $1 \le n \le 100$.

Example

Sample Input	Sample Output
3 72 100 35	195 217 48

Problem B

Interior Angles of a Polygon

(Time Limit: 5 seconds)

An interior angle is an angle located inside a polygon, and it lies between two adjacent edges of a polygon. A simple polygon consists of finitely distinct vertices and many straight-line edges, each vertex in it is exactly shared by two edges, and the edges are non-intersecting. A simple polygon is regular if it has equal angles and equal edges, or else the polygon is irregular. Furthermore, a simple polygon is either convex or concave. For a polygon to be convex, all of its interior angles must be less than or equal to 180 degrees; otherwise, the polygon is concave.

A shopping center wants to install the modern Internet Protocol(IP) network camera system on the first floor to improve the surveillance and security services of the center. The security department decides to choose the Infrared IP network camera for indoor use with 30-meter night vision, and places the cameras in all corners of the floor to monitor inside the surroundings of the shopping center. We know that the maximum viewing angle of each camera is 45 degrees, and your goal is to determine the minimum number of total cameras that suffice to fully cover the entire corners of the floor in any case.

As an example, if an interior angle of a corner is 90.01 degrees, we need to install 3 IP cameras for covering the entire corner.

Input Format

The first line of the input gives the number of test cases, k ($1 \le k \le 10$), and k cases follow. The first line of each test case contains an integer n ($3 \le n \le 10$), the number of corner pairs bounding the shopping center. Each of the following n lines has two integers, x followed by y ($|x|, |y| \le 10^3$), indicating the coordinates (x, y) of the shopping center's corner locations in clockwise order. The aerial view of the shopping center shows that its plane figure's shape is a simple polygon.

Output Format

Display the minimum number of total IP cameras that suffice to cover the corners of the first floor of the shopping center.

Technical Specification

- k is an integer, $1 \le k \le 10$.

Problem B

- n is an integer, $3 \le n \le 10$.
- x and y are integers, $|x|, |y| \le 10^3$.

Example

Sample Input	Sample Output

3
5
0 0
1 1
2 2
3 3
4 0
5
10 10
0 0
10 -10
-10 -10
-10 10
7
0 0
5 5
10 0
6 0
6 -5
0 -5
4 0
	13
	12
	21

Problem C
Person Identification System using RFID Technology

(Time Limit: 1 seconds)

Due to the threat of Covid-19, a professor developed a person identification system using RFID technology to identify the persons entering the buildings of his university. As shown in the below figure, the person who wants to enter a building needs to sweep his/her student ID card or employee ID card over an RFID reader. The Unique Identifier (UID) of the card is read by the reader and the 4 bytes' hexadecimal (shortly hex) reverse order with leading zeros (always 8 hex digits) of the UID is searched in a simple off-line database. If one record could be found, the student ID or employee ID in the record is displayed on the screen of a notebook. Otherwise, an uppercase letter, X, is displayed, instead. Note that the reader reads the UID into a decimal number, a total of 10 digits. Besides, the byte order of the hex for the decimal number will be inverted for the following database search.

Take UID, ***0123456789***, as an example. The hex value of 0123456789 is 075BCD15. The reverse hex value is 15CD5B07, which is searched in the database. Now, the professor hopes you reproduce his system.

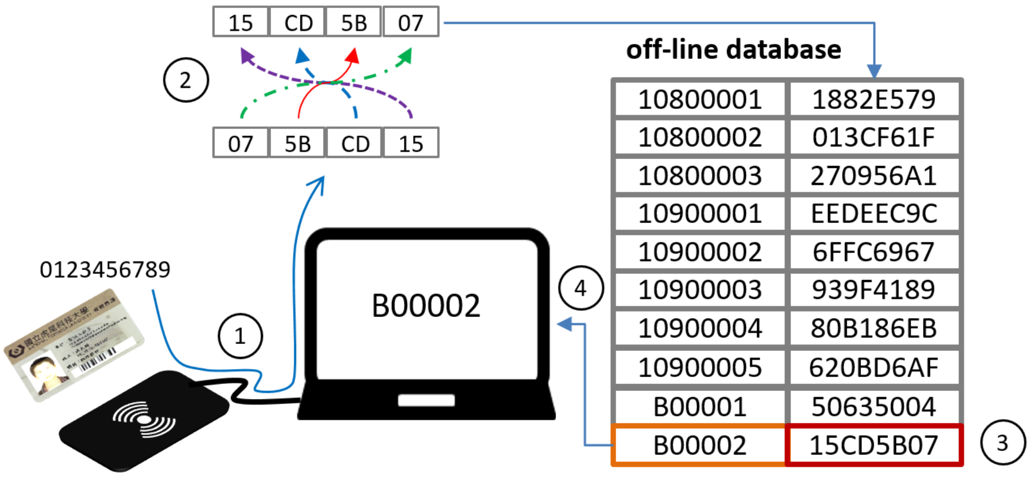


Fig. 1. Data flow chart.

Input Format

The first line gives the number of records in the off-line database, then followed by the lines for the input data of each record. For each record, there is a line with a string consisting of two fields separated by a comma. The first field can be either a student ID, a total of 8 decimal digits, or an employee ID, a letter of the alphabet followed by 5 decimal digits. The second field is the four bytes of UID in reverse order. After this, there comes a single positive integer n, which specifies the number of test cases. Each of the next n lines consists of a UID composed by 10 decimal digits.

Problem C

Output Format

For each test case, output the search result in one line. If one record could be found in the database, the search result will be the student ID or employee ID in the record. Otherwise, the search result will be an uppercase letter, X.

Technical Specification

- There are at most **12000** records and **6000** test cases.
- Each student ID or employee ID is unique.
- Each RFID card, including the student ID card and employee ID card, has a unique UID.
- The UID ranges from 0000000000 to 4294967295. In other words, it ranges from 0x00000000 to 0xFFFFFFFF.

Example

Sample Input	Sample Output
10 10800001,1882E579 10800002,013CF61F 10800003,270956A1 10900001,EEDEEC9C 10900002,6FFC6967 10900003,939F4189 10900004,80B186EB 10900005,620BD6AF B00001,50635004 B00002,15CD5B07 5 0123456789 0123456789 0313145542 0536230913 2632769262	B00002 B00002 X 10800002 10900001

Problem D
Lollipops

(Time Limit: 5 seconds)

Elementary school student Johnson likes to share lollipops with his friends, but he always distributes lollipops unevenly. Johnson's friend Jenny told Johnson that lollipops should be distributed fairly among friends. Johnson thought she was right, so he decided to move some lollipops so that his friends could get as many lollipops. Since Johnson is a lazy kid, he wants to move the smallest number of lollipops so that he can distribute the lollipops fairly to his friends.

Can you write a program to help him? The program needs to take the following steps.

Step 1: Find the lollipop pile A satisfying the condition $n_a < n_{avg}$ from the left in order.

Step 2: Find one lollipop pile B satisfying the condition $n_b > n_{avg}$ from all the piles, and from the pile B move n lollipops to pile A at once, repeat this step until n_a is equal to n_{avg}.

Step 3: Repeat Step 1 and 2 until all the piles have the same number of lollipops.

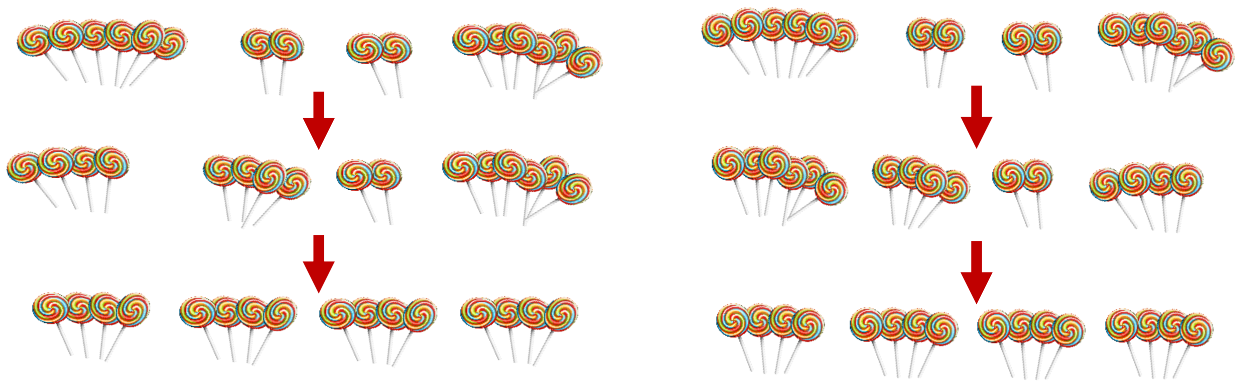
n_{avg} : The average number of lollipops of all piles.

n_a : The number of lollipops of pile A.

n_b : The number of lollipops of pile B.

n : The smaller number of $(n_{b-}n_{avg})$ and $(n_{avg} - n_a)$.

Please write a program to calculate the minimum number k of moving lollipops to make the number all the piles the same, and the number of moving methods m with the number of moves k. The following figure shows the moves of lollipops.



Problem D

Input Format

The first line of input has a positive integer L $(1 \leq L \leq 7)$ indicating the number of test cases. The first line of each test case has a positive integer x indicating the number of the lollipop piles. The second line of each test case has x positive integers $(n_i\ , 1 \leq i \leq x)$ representing the lollipops number of each lollipop pile, where $1 \leq x \leq 12$, $1 \leq n_i \leq 50$. The total number of lollipops of each test case must be divisible by the number x, which means that the lollipops can be distributed evenly.

Output Format

Print the number of the least moves k and the number of k-move methods m for each test case in a line.

Example

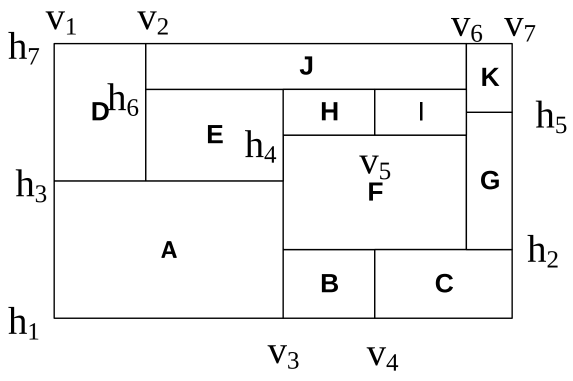
Sample Input	Sample Output

2
4
6 2 2 6
4
7 2 2 5
	2 2
	3 3

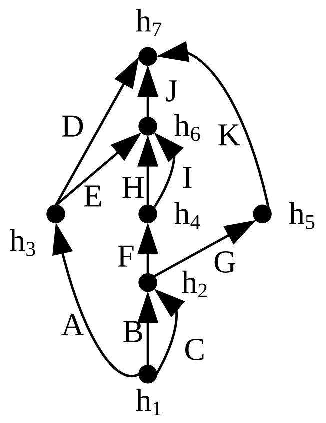
Problem E
Coordinates of Constituting Rectangles

(Time Limit: 5 seconds)

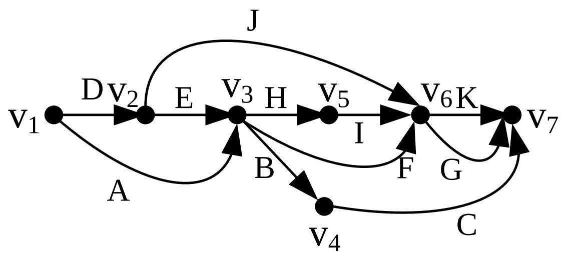
Given a big rectangle which is formed by assembling some constituting rectangles as shown in Fig. 1(a), we can use two graphs to denote the positional relationship of the constituting rectangles. There is no overlap and also no empty space between any two constituting rectangles. The first graph is called *vertical polar graph* as shown in Fig. 1(b), which consists of a set of nodes and directed edges. A node represents a line segment in the big rectangle. A directed edge from a node h_1 to a node h_3 as shown in Fig. 1(b) denotes that the line segment h_1 can reach h_3 by traversing the constituting rectangle **A**. A line segment in the big rectangle is formed by joining some same-direction edges of constituting rectangles end to end. For example, line segment h_1 is formed by joining the three horizontal edges at the bottom of rectangles **A**, **B**, and **C**, respectively. No line segment can contain another line segment. A horizontal polar graph as shown in Fig. 1(c) can be constructed in the same way. Now, given the two polar graphs and the dimensions of the constituting rectangles of a big rectangle, the problem is to determine the coordinates of the lower left corner of each constituting rectangle. Here, it is assumed that the *y* coordinate of the bottom most horizontal line segment such as h_1 in Fig.1 is zero and the *x* coordinate of the left most vertical line segment



(a). Constituting rectangles



(b). Vertical polar graph



(c). Horizontal polar graph

A:(10,6) B:(4,3)
C:(6,3) D:(4,6)
E:(6,4) F:(8,5)
G:(2,6) H:(4,2)
I:(4,2) J:(14,2)
K:(2,3) (width, height)

(d). Width and height of rectangles

Fig. 1. Polar graphs for representing positional relationship among constituting rectangles.

Problem E

such as v_1 is zero. As an example, given the dimensions of the constituting rectangles in Fig. 1(d), the coordinates of the lower left corner of the constituting rectangles are A=(0, 0), B=(10, 0), C=(14, 0), D=(0, 6), E=(4, 6), F=(10, 3), G=(18, 3), H=(10, 8), I=(14, 8), J=(4, 10), and K=(18, 9).

Input Format

The first line gives the number of test cases. Each test case has three sections of data. The first section gives the dimensions of the constituting rectangles. It starts with a line consisting of "RECT *n*", where RECT is a keyword and *n* is the number of constituting rectangles. It is then followed by *n* lines where each line has a string and two integers in sequence. They denote the name, width, and height of a constituting rectangle, respectively. The second section starts with a line consisting of a keyword "VPG". It is then followed by *n* lines where each line has three fields F1, F2, and F3 specifying a directed edge in a vertical polar graph. F1 is an integer denoting the starting node of a directed edge and F3 is also an integer denoting the ending node. F2 gives the name of a constituting rectangle being traversed to form the directed edge. The third section specifies a horizontal polar graph. It starts with a line consisting of "HPG". The input data of a horizontal polar graph and a vertical graph are presented in the same way.

Output Format

The output for each test case has *n*+1 lines where *n* is the number of constituting rectangles in a big rectangle. The first line consists of "TEST *x*:" where *x* should be replaced by the test case number. Hereon, each line gives the name of a constituting rectangle and the *x* and *y* coordinates of its lower left corner. The output lines for constituting rectangles should be sorted in ascending lexicographic order in terms of their names, i.e., the order determined by string comparison operator < with smaller being printed first.

Technical Specification

- There are at least two but no more than 10000 constituting rectangles in a big rectangle.
- The number of test cases is not more than 10.
- The coordinates and dimensions of a constituting rectangle are representable in 32-bit non-negative integers.
- The name of constituting rectangles consists of only English alphabets and digits.

Problem E

Example

Sample Input	Sample Output
2	TEST 1:
RECT 11	A 0 0
B 4 3	B 10 0
C 6 3	C 14 0
D 4 6	D 0 6
E 6 4	E 4 6
F 8 5	F 10 3
G 2 6	G 18 3
H 4 2	H 10 8
I 4 2	I 14 8
J 14 2	J 4 10
K 2 3	K 18 9
A 10 6	TEST 2:
VPG	A 0 0
1 A 3	D 0 6
1 B 2	E 4 6
1 C 2	J 4 10
2 G 5	
2 F 4	
3 D 7	
3 E 6	
4 H 6	
4 I 6	
5 K 7	
6 J 7	
HPG	
1 D 2	
1 A 3	
2 E 3	
2 J 6	
3 B 4	
3 H 5	
3 F 6	

Problem E

```
4 C 7
5 I 6
6 G 7
6 K 7
RECT 4
D 4 6
E 6 4
A 10 6
J 6 2
VPG
1 A 3
3 D 7
3 E 6
6 J 7
HPG
1 A 3
1 D 2
2 E 3
2 J 3
```

Problem F
Nuclear Crisis

(Time Limit: 3 seconds)

A nuclear crisis happens now at Formosa Nuclear Power Plant. You are asked to immediately calculate the scale of released energy such that scientists can determine which method to save the power plant. According to nuclear scientists, the released energy obeys the following formula. That means the nuclear plant released $f(n)$ joules at n microseconds after the crisis happens. $f(0) = 2$, $f(1) = 4 + 2\sqrt{2}$ are given, and for $n \geq 2$, we have

$$f(n) = (2 + 2\,cos\left(\frac{n\pi}{4}\right)) \times f(n-1) + (2 + 2\,sin\left(\frac{n\pi}{4}\right)) \times f(n-2) + (1 + cos(n\pi)) \times 2^n$$

For example, $f(2) = (2+0) \times f(1) + (2+2) \times f(0) + (1+1) \times 2^2 = 2 \times (4 + 2\sqrt{2}) + 4 \times (2) + 2 \times 2^2 = 24 + 4\sqrt{2}$, and $f(3) = (2-\sqrt{2}) \times f(2) + (2+\sqrt{2}) \times f(1) + (1-1) \times 2^3 = (2-\sqrt{2}) \times (24 + 4\sqrt{2}) + (2+\sqrt{2}) \times (4 + 2\sqrt{2}) = 40 - 16\sqrt{2} + 12 + 8\sqrt{2} = 52 - 8\sqrt{2}$.

It is trivial that $f(n)$ can be represented as $u + v \times \sqrt{2}$ where u and v are integers. In order to check the correctness of your calculation easily, please output two integers $x = (u \ mod \ M)$ and $y = (v \ mod \ M)$ where $0 \leq x < M$, $0 \leq y < M$ and M is given as 20210707. Take n=3 as an example, $f(3) = 52 - 8\sqrt{2}$, you have to output 52 and 20210699.

Input Format

The first line has the number of test cases. Each test case has an integer n.

Output Format

For each test case, please output the corresponding values x and y for $f(n)$.

Technical Specification

- There are at most 10 test cases.
- $1 \leq n \leq 10^{202}$.

Problem F

Example

Sample Input:	Sample Output:

5
0
1
2
3
4
	2 0
	4 2
	24 4
	52 20210699
	80 8

Problem G
Sum of Pure Infinitely-repeating Decimal Numbers

(Time Limit: 1 seconds)

A decimal number in which all the digits of the floating-point part are nonzero and repeat infinitely is called a *pure infinitely-repeating decimal number*. For example, $3.\overline{142857}$ is a pure infinitely-repeating decimal number, whose *repetend* is 142857; $3.6\overline{142857}$ is not a pure infinitely-repeating decimal number because it contains non-repeating digit in its floating-point part. In this problem, all decimal numbers are pure infinitely-repeating decimal numbers. Please write a program to compute the sum of two pure infinitely-repeating decimal numbers.

Input Format

The first line is an integer n indicating the number of test cases. Each test case consists of two pure infinitely-repeating decimal numbers x and y. Both x and y may be positive or negative. For example, 2.3 indicates $2.\overline{3}$; 1.23 indicates $1.\overline{23}$.

Output Format

For each test case consisting of pure infinitely-repeating decimal numbers x and y, print the sum of x and y. Note that your output may either a pure infinitely-repeating decimal number or an integer. Even though $0.14\overline{285714}$ and $0.\overline{142857}$ are numerically equal, the former is not allowed. For $0.\overline{9}$, your program should output an integer, rather than a decimal.

Technical Specification

- Each test case consists of two decimal numbers x and y from (–101, 101), i.e., $-101< x <101$ and $-101< y <101$. The repetends of both x and y will not exceed 9 digits.

Example

Sample Input	Sample Output
8	12.78
9.2 3.56	3.577668
2.34 1.234	11.111400329222
9.876 1.2345	7.031746
6.8 0.142857	29.49

Problem G

33.82 -4.3
-78.3523 30.1888
1.234 4.321
88.3 11.6
	-48.1635
	5.5
	100

Problem H
The Densest Cuboid of a 3D matrix

(Time Limit: 1 seconds)

Given an $n \times n \times n$ 3D-matrix $A = [a_{i,j,k}]$, and a positive integer L, the objective of the problem is to find a (rectangular box) *cuboid*, a (consecutive) sub-(3D)-matrix of A, with each dimension of size *at least* L such that the *average* of the elements in the cuboid is maximized. Note that the size (number of elements) of the cuboid is at least L^3 but can be much larger than L^3.

The value of each element $A = [a_{i,j,k}]$, is defined by 3 length-n sequences, $F = \langle f_i \rangle_{i=1}^{n}$, $G = \langle g_i \rangle_{i=1}^{n}$, and $H = \langle h_i \rangle_{i=1}^{n}$ with the following formula

$$a_{i,j,k} = f_i^2 + f_j^2 + 2(f_i + f_j) + 3(g_i^2 + g_k^2) + 4(g_i + g_k) + 5(h_j^2 + h_k^2) + 6(h_j + h_k)$$

The *density* of an $\ell \times m \times p$ cuboid (sub-3D-matrix) $B = [b_{i,j,k}]$ is the average value of all elements of B. That is, $(\sum_{i=1}^{\ell} \sum_{j=1}^{m} \sum_{k=1}^{p} b_{i,j,k})/(\ell m p)$. Note that finding the densest cuboid of a 3D-matrix is trivial. It is just the largest element within the matrix. On the other hand, the problem becomes interesting when we are asked to find the cuboid with each dimension of length at least L such that the density of the cuboid is maximized.

Write a program to find the densest cuboid of A with each diminsion of size *at least* L.

Input Format

The first line gives the number of test cases. For each test case, the first 2 integers, namely, n and L, represent the size of the 3D-matrix is n^3 and the size of the constrained cuboid with each diminsion of size *at least* L. Note that $1 \le L \le n \le$ 300,000. Following n and L, the rest of the input are 3 length-n integer sequences, namely F, G, H. For elements of these sequences, we have $0 \le f_i, g_i, h_i \le 10^3$.

That is, each set of the test case are just $2 + 3n$ integers. These test cases will occur repeatedly in the input as the pattern described above.

Output Format

For each matrix of the input, find the densest cuboid of A with each diminsion of size *at least* L. Output the density of the densest sub-cuboid. Print the results with 4 digits to the right of the decimal point. That is, with the "%.4f" precision specifiers if the program is written in C printf function; in

Problem H

Python, with command print("{:.4f}".format(val))； in Java, with satement System.out.printf("%.4f", val);

Technical Specification

- $1 \le L \le n \le 300{,}000$, and $0 \le f_i, g_i, h_i \le 10^3$.

Example

Sample Input	Sample Output

2
1 1
1
1
1
4 2
78 98 69 54
13 28 63 38
12 19 64 52
	42.0000
	61640.0000

Problem I

Binomial theorem

(Time Limit: 3 seconds)

The binomial theorem expands expression of $(a + b)^n$ as following expression.

$$(a+b)^n = \binom{n}{0} a^n b^0 + \binom{n}{1} a^{n-1} b^1 + \binom{n}{2} a^{n-2} b^2 + \cdots + \binom{n}{n} a^0 b^n$$

$$where \qquad \binom{n}{r} = {}^nC_r = \frac{n!}{r!\,(n-r)!}$$

The binomial coefficients $\binom{n}{r}$ for varying n and r can be arranged to form Pascal's triangle shown as Fig. 1,

```
              1
            1   1
          1   2   1
        1   3   3   1
      1   4   6   4   1
    1   5  10  10   5   1
  1   6  15  20  15   6   1
1   7  21  35  35  21   7   1
```

Fig. 1 Pascal's triangle for $n = 7$

Write a program to print out Pascal's triangle based on binomial theorem for $n \leq 10$.

Input Format

The first line is an integer indicating the number (≤ 10) of cases. For each case, the input n is an integer.

Output Format

For each test case, the output is a Pascal's triangle.

Technical Specification

Problem I

- 0≤n≤10.

Example

Sample Input:	Sample Output:
3	1
0	
2	1
7	1 1
	1 2 1
	1
	1 1
	1 2 1
	1 3 3 1
	1 4 6 4 1
	1 5 10 10 5 1
	1 6 15 20 15 6 1
	1 7 21 35 35 21 7 1