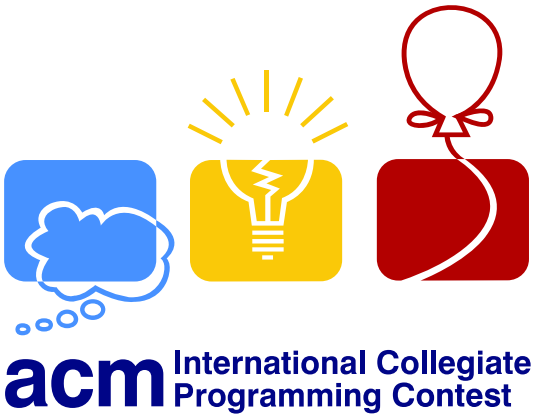
South German Winter Contest 2011

February 05, 2011



The Problem Set

Good luck and have fun!

This page is intentionally left (almost) blank.

Problem A
Age of the Universe

Recently, a clever researcher proposed a formula for the expansion of the universe as a function of the age t of the universe:

$$expansion(t) = \sqrt[a]{b \cdot t^c \cdot log_{1+d}(1 + t^e) \cdot exp(t^f)}$$

In order to verify his formula, he gives you the values for the coefficients and the current expansion. Can you calculate the age of the universe?

Input

The first line of the input gives the number of test cases C ($0 < C < 100$). Each of the test cases is described with seven integers on one line. The numbers are given in the following order: $expansion(t), a, b, c, d, e, f$ where you may safely assume $1 < expansion(t) < 10\,000$ and $0 < a, b, c, d, e, f \leq 5$.

Output

For each test case, print one line containing the estimated age of the universe for the given coefficients and expansion. You may safely assume that $0 \leq t \leq 42$.
Your output should have an error of at most 10^{-4}.

Sample Input	Sample Output
5	0.891631746526
2 1 1 1 1 1 1	2.877690715650
100 1 1 1 1 1 1	40.663063754437
9999 5 1 1 1 1 1	1.508742619080
42 1 2 3 4 5 1	2.009768252719
23 3 2 1 5 2 3	

This page is intentionally left (almost) blank.

Problem B

Bazinga!

When Sheldon fools someone (in the majority of cases Leonard is his victim), he uses the term *Bazinga!* as a follow up to his jokes or pranks. Usually, he does not see Sheldon's jokes coming, but this time Leonard found a paper with a Java class named "Bazinga" on it. He guesses that Sheldon prepares another prank however Leonard is not good with programming languages.

Sheldon's Bazinga! class

```
import java.util.*;

public class Bazinga {
        public static void help(int[] numbers, int pos) {
                Stack<Integer> stack = new Stack<Integer>();
                for (int i = 0; i < pos; i++) {
                        stack.push(numbers[i]);
                }
                for (int i = 0; i < pos; i++) {
                        numbers[i] = stack.pop();
                }
        }

        public static void process(int[] numbers, int len) {
                for (int i = len; i > 1; i--) {
                        int pos = 0;
                        for (int a = 0; a < i; a++) {
                                if (numbers[pos] < numbers[a]) {
                                        pos = a;
                                }
                        }
                        if (pos == i-1) continue;

                        if (pos > 0) help(numbers, pos + 1);

                        help(numbers, i);
                }
        }

        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                int numCases = sc.nextInt();

                for (int i = 0; i < numCases; i++) {
                        int numNumbers = sc.nextInt();
                        int numbers[] = new int[numNumbers];

                        for (int j = 0; j < numNumbers; j++) {
                                numbers[j] = sc.nextInt();
                        }

                        process(numbers, numbers.length);

                        for (int j = 0; j < numNumbers - 1; j++) {
                                System.out.print(numbers[j] + " ");
                        }
                        System.out.println(numbers[numNumbers - 1]);
                }
        }
}
```

Leonard tried to compile and run the program but it seems to be terribly slow. Can you help him to write a faster variant?

Input

Leonard already found out that some test cases are given as input. The first line of the input gives the number of test cases C ($0 < C < 100$). Each of the test cases is given on two lines: the first line of each test case specifies the number N of numbers that are processed by the `process` method in Sheldon's code ($0 < N < 100\,000$). The second line of each test case gives N numbers, each number is positive and at most $2^{31} - 1$.

Output

For each test case, your program should print one line containing the same numbers as they would have been printed by Sheldon's code.

(There is no Sample Input or Output provided for this problem.)

This page is intentionally left (almost) blank.

Problem C

Big Bang Research

Doing research about the big bang – which led to the formation of the universe – can be painful due to the large scales in space. The area we want to investigate should be as small as possible. Therefore, we split the universe into a grid and are only interested in the most promising of the resulting cells.

You are given a couple of cuboids that are axis-aligned with respect to the grid. Each of these cuboids has an interest value assigned to it (higher is better). If cuboids intersect, the interest value for each grid cell in the intersected volume is calculated by adding the values of the corresponding cuboids.

Given a set of cuboids, can you tell us which cell of the grid is the most promising one?

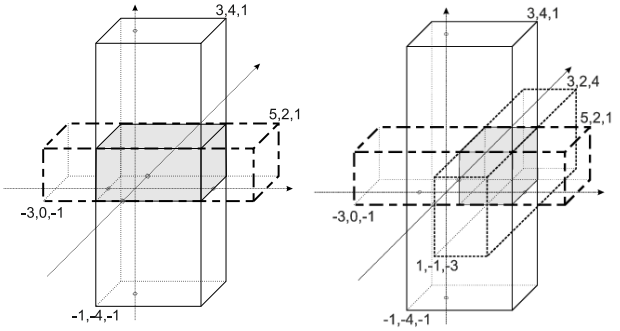


Figure 1 – Cuboids from second and third sample input. Overlapping volumes are shaded. In the example on the left, there are 16 grid cells each of which has an interest value of $12 + 3 = 15$. You have to select the cell with the lowest coordinates: $(-1, 0, -1)$. Cells are identified by their lower left corner.

Input

The first line of the input gives the number of test cases t ($0 < t < 100$). Each of the test cases is given as follows: the first line of each case gives the number of cuboids C ($0 < C < 25$). Then follow C lines, first giving the three-dimensional coordinates for two opposing corners of each cuboid, then its interest value. The coordinates are specified with integers in the interval $-1\,000\,000 < x_i, y_i, z_i < 1\,000\,000$. Each coordinate for the first corner is lower than the corresponding coordinate for the opposing corner. The interest value is in the interval $0 < i < 1\,000$.

Output

For each test case, print one line containing the identifying coordinates of the most promising grid cell and the corresponding interest value. If there are several of them, print the cell with the lowest x-coordinate. If there are two with the same x-coordinate, print the cell with the lowest y-coordinate. If there is still a tie, print the one with the lowest z-coordinate.

Sample Input

```
4
1
-10 -10 -10 10 10 10 42
2
-1 -4 -1 3 4 1 12
-3 0 -1 5 2 1 3
3
-1 -4 -1 3 4 1 12
-3 0 -1 5 2 1 3
1 -1 -3 3 2 4 8
5
-1 -4 -1 3 4 1 12
-3 0 -1 5 2 1 3
1 -1 -3 3 2 4 8
-999999 -999999 -999999 999999 999999 999999 815
-1234 -1234 -1234 -1233 -1233 -1233 815
```

Sample Output

```
-10 -10 -10 42
-1 0 -1 15
1 0 -1 23
-1234 -1234 -1234 1630
```

This page is intentionally left (almost) blank.

Problem D
Fuel Revolution

Probably you remember that Leonard was working very hard on a rocket fuel for the government.[1] Sheldon and Leonard use the research results to invent a new kind of car fuel. Producing this fuel is easy and cheap. Furthermore, the cars that consume this fuel are almost as fast as light. But there is one problem they have to face when handling this new resource: The fuel tank has to be very safe because the fuel is highly explosive (remember the elevator). Therefore, the new fuel tank has to be as small as possible.

Thus, Sheldon and Leonard want to know how large their car fuel tank has to be to be useful in practice. While driving to their destinations, Sheldon and Leonard can refuel arbitrarily often at fuel stations. In order to calculate the tank's size, they want to know how far they can go on a trip without refueling. They ask you to calculate the maximal distance between two fuel stations. Of course, they choose optimal paths to minimize this maximal distance and hence the tank's size. Sheldon and Leonard always start and end at their favorite fuel station.

Input

The first line of the input gives the number of test cases C ($0 < C < 100$). The first line of each such test case holds N, M, T, and Q: the number of nodes in the map ($2 \leq N \leq 10\,000$), the number of roads in the map ($0 \leq M \leq 50\,000$), the number of fuel stations ($1 \leq T \leq 20$), and the number of queries ($0 \leq Q \leq 5\,000$). Each of the following M lines holds three integers n_1, n_2, and c that describe a road. Each line specifies one one-way-road from n_1 to n_2 with cost c ($0 \leq n_1, n_2 < N$, $n_1 \neq n_2$, $0 < c \leq 1\,000$). It is assured that for every two nodes n_1 and n_2 there is at most one road from n_1 to n_2. The next line has T integers t_i ($0 \leq t_i < N$) that are nodes with fuel stations – the first fuel station in this list (t_1) is the favorite fuel station of Sheldon and Leonard. Each of the following Q lines gives an integer q_i ($0 \leq q_i < N$, $q_i \neq t_1$) that specifies a possible destination for Sheldon and Leonard.

Output

For each test case print $Q + 1$ lines. The first line should be "`Case` C`:`" where C is the index of the test case. Each of the following Q lines should give one integer: the best minimal distance between any two fuel stations along all the potential routes from their favorite fuel station to node q_i and back. If it is not possible at all to reach the destination and return to the start node, print "`IMPOSSIBLE`".

Sample Input

```
1
7 8 5 6
0 1 5
1 2 5
2 6 5
2 3 4
3 0 7
3 4 3
5 0 17
6 0 10
0 1 5 4 6
1
2
3
4
5
6
```

Sample Output

```
Case 1:
10
10
16
IMPOSSIBLE
IMPOSSIBLE
10
```

[1] When experimenting with Howard's rocket, Leonard's math caused the rocket to explode in an elevator.

This page is intentionally left (almost) blank.

Problem E

How many Universes?

God is bored. So he decides that something must be created. In his sandbox he starts testing out elements. He creates a certain number of elements, e.g. iron, gold, hydrogen, and what else he can think of. But pure elements are boring too, so he thinks about combining them to create big bangs and corresponding universes. He wonders how many different universes he can create from a certain number of elements. So, e.g. for his first test, he creates five elements. When using every possible combination of four of his five elements, he can create five big bangs and thus five different universes. When creating universes the order in which the elements are selected does not matter. Also one element cannot be taken twice. When he takes three of his five elements, he can create ten universes. With all five elements just one universe can be created.
God discovers that universes based on just a little number of elements do not give much excitement and do not last long. Also he wants to have more choices so he creates more and more elements and picks more elements from them. Imagine at one point he even creates 55 different elements and starts a whole bunch of universes in parallel because he really likes the game. With all 55 elements he can again just create one universe, but when just taking 33 of 55 he can learn about the fate of $1\,300\,853\,625\,660\,225$ universes.
You live in a universe with 118 elements. Usually, in universes of this size, some kind of life forms exist, that is able to handle numbers and calculations. So God is testing them and you. He gives you his current number of created elements and he tells you how many elements thereof he is going to take for the creation of universes. You must calculate how many parallel universes he can create. And you should calculate fast otherwise your universe will be terminated early because it is no fun to watch.

Input

God is gracious, so the maximum number of elements he will be using for your test is 55. In his test definition, he will first give you the number of different tests C ($0 < C < 2\,000$). Each test consists of two numbers: the number of all possible elements E and the number of elements T taken from these for creating universes ($1 \leq E \leq 55$, $1 \leq T \leq E$).

Output

Your task is to calculate the number of possible universes for each test case. Just output this number, each number should be written in a new line.

Sample Input	Sample Output
6	1
5 5	5
5 4	10
5 3	10
5 2	1300853625660225
55 33	3824345300380220
55 27	

This page is intentionally left (almost) blank.

Problem F

Inclined plane

As an experimental physicist, Leonard has a lot of inclined planes (ramps) each of which is one inch wide. He wants to store them in a square area. To save space, he rotates the ramps in a way that they stand on their smallest side. To be able to play with his matchbox cars on the ramps he puts two ramps in a row, so that they are connected at their lowest points. To get his square of ramps financed he sets up an experiment that substitutes parts of his ramps in the array with ice replications. He needs many of these ice replications and asks you to calculate how many water he needs for the substitutes.
Leonard uses the coordinates of an axis-aligned rectangle to specify which part of the ramp array has to be replaced by ice substitutes, namely exactly those parts of ramps that stand on the specified floor area. The substitute ice blocks should have the same surface as the original ramps. An example of an array is given in the figure. The different densities of ice and water can be ignored as measuring inaccuracy.

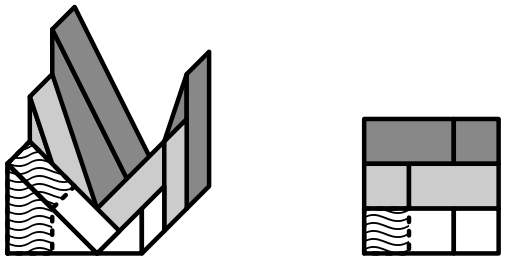


Figure 2 – Array of 2nd test case with 1st query

Input

The input starts with the number of test cases C in one line ($0 < C < 100$). Each test case starts with a line consisting of the number of rows n. Then n lines follow, each describing one row of the ramp array, starting with the frontmost row. Each description of a row consists of three integers: The length l of the left ramp, the slope s_1 of the left ramp in height / length and the slope s_2 of the right ramp in the same notation. The next line gives the number of queries q, followed by q lines each describing one query. Each query specifies one rectangle of the floor of the ramps array that needs to have its ramps / ramp segments substituted. The four integers specify the front left corner followed by the back right corner of the rectangle ($(0, 0)$ is front left).
Consider the following limits: $0 < n, s_1, s_2, q \leq 40\,000$ and $0 < l < n$.

Output

The volume under the ramps for each query in inch^3 with two digits after the point.

Sample Input	Sample Output
2 4 2 1 1 2 1 1 2 1 1 2 1 1 2 0 0 4 4 1 1 3 3 3 2 1 1 1 3 1 2 2 3 2 0 0 1 1 2 0 3 1	16.00 2.00 1.50 0.50

This page is intentionally left (almost) blank.

Problem G

Language Confusion

Leonard cannot stand Howard for using foreign languages — especially when hitting on Penny. He collected word lists for the different languages Howard uses. So at least he can call out phrases like "Oh, you're talking in French again. Feeling very cool, yeah?". But looking up the words in a notebook is far too slow. You are supposed to help Leonard with this issue.

Input

The first line contains the number of test cases C ($0 < C < 100$). Each test case starts with a line that holds the number n of languages Howard uses ($0 < n \leq 5$). Then n blocks follow.
The i-th block describes the i-th language. Its first line contains the name of the language and the number of words c_i ($1 \leq c_i \leq 100$) known to Howard in that language. Then follow the c_i words that make up the word list, each on a single line.
After the blocks there is the description of the queries. The first line contains the number of queries q ($0 < q \leq 20$). Each of the following q lines contains a single word.
The words and language names only contain the letters a-z and A-Z and have at most 30 characters each. The case of the letters does not matter for the comparison as you can see in the sample input.

Output

For each query, print the name of the language to which the word belongs (according to the word lists). If there are multiple possible languages, print "`Ambiguous`". If the word does not appear in any of the lists, print "`Unknown`".

Sample Input

```
2
3
English 5
test
girl
and
see
computer
German 3
test
Auto
See
Italian 4
Compagnio
lago
Compagnio
antico
3
compagnio
Test
hand
2
Suomi 2
LAGO
kieli
English 1
nothing
3
lAgO
girl
compagnio
```

Sample Output

```
Italian
Ambiguous
Unknown
Suomi
Unknown
Unknown
```

This page is intentionally left (almost) blank.

Problem H
Next Big Bang

As everybody knows, it is assumed that there was a big bang in the past. Since then, the universe is extending. The question is whether the gravitation between different galaxies is strong enough to stop the expansion at some point in the future. Several physicists work on this issue and performed a few measurements. They wrote down the coordinates of N galaxies at two different times (some years in the past and now).

To simplify their model, the physicists assume that the galaxies move linearly.[2] An important value in the model is the "*safe factor*". The safe factor of two galaxies is calculated by dividing d_{min} by d_{cur} whereby d_{cur} is the current distance between the two galaxies and d_{min} is the minimal distance in the future.

If two galaxies are striding away from each other from now on (i.e. the distance is increasing), the safe factor is one. On the other hand, if two galaxies will crash in the future (i.e. the distance is zero at some point), the safe factor is zero. The safe factor over all given galaxies is the minimal safe factor of all pairs of galaxies. You can assume that no two galaxies are currently at the same position. As the physicists are busy doing additional measurements they ask you to calculate the safe factors for some systems of galaxies. Based on your results, the physicists will decide whether it is possible that some galaxies will crash or not.

Input

The first line of the input provides the number of test cases C ($0 < C \leq 100$). The first line of each test case holds the integer N ($2 \leq N \leq 50$). Each of the following N lines contains six integers $xp_i, yp_i, zp_i, xc_i, yc_i$ and zc_i ($-10^6 \leq xp_i, yp_i, zp_i, xc_i, yc_i, zc_i \leq 10^6$). The first three integers specify the position of the i-th galaxy in the past and the second three integers specify the current position of the i-th galaxy.

Output

For each test case print one line of output containing the safe factor of the corresponding system of galaxies. Your output should have an error of at most 10^{-5}.

Sample Input

```
4
3
0 0 0 2 0 0
0 0 0 0 2 0
0 0 0 0 0 2
2
0 1 1 0 2 1
0 0 0 0 1 0
2
0 0 0 0 0 0
0 -2 1 0 -1 1
4
1 2 3 4 5 6
7 8 9 0 0 0
-500 -500 -500 -250 -250 -250
-20 -23 -42 8 15 4711
```

Sample Output

```
1.0000000000
1.0000000000
0.7071067812
0.0031646996
```

[2] In fact, the galaxy movement is non-linear because of the gravitation but we neglect this for this estimation.

This page is intentionally left (almost) blank.

Problem I

Penny's Business

Penny is broke – once again. In need of money, she remembers the time when she founded a business and sold *Penny Blossoms*. She decides to start up her business again. Because of the disaster when she last tried it, she cannot ask one of the boys for help. Thus, she has to do it all on her own. She needs 24 minutes to create one Penny Blossom. Unlike last time no one sets up an online shop for her, so she takes offers by mail. All orders come in on the first day of the month, so she can plan one month ahead. This allows her to also refuse orders if she will not be able to fulfill them. Hard-working as she is, she does not want to work more than eight hours per day, but will also work on holidays and on the weekends.
Help Penny to choose which orders to take. She only takes orders she can fulfill completely. Her goal is to accept as many orders as possible in order to increase her customer base.

Input

Input starts with a positive number, the number of test cases (or months) C, on a single line ($0 < C \leq 100$). After that, a single line per test case follows. The line starts with n, the number of orders, followed by n pairs of numbers that specify both the day of the month d_i on which the order is due (it is sufficient to have the blossoms ready at the end of that day) and the number a_i of ordered blossoms ($0 < n < 2\,000$, $1 \leq d_i \leq 30$, $1 \leq a_i \leq 600$).

Output

Output a single line per test case, containing the maximal number of orders that Penny can accept in the corresponding month.

Sample Input	Sample Output
3	3
3 2 20 3 20 1 20	3
4 1 4 2 42 3 50 4 20	5
6 20 47 11 42 23 7 3 30 2 40 3 10	

This page is intentionally left (almost) blank.

Problem J
Sheldon's Toy Soldiers

When he was a kid Sheldon started collecting toy soldiers. Compulsive as he is, he has a strict principle for lining them up in a single line on his shelf in the same order he bought them. Sometimes he wants to select soldiers to form a detachment. The detachment has to be ordered from the tallest to the smallest toy soldier. To save time (you have to be quick when there is war!), Sheldon only allows himself to walk once along the shelf and select soldiers in the order they appear on the shelf. But to ease the process he also needs them to be picked so that after picking they are lined up from tallest to smallest. On each pick, he can only select a soldier that is smaller than all the soldier picked before. Given a description of the soldiers on Sheldon's shelf, what is the largest number of soldiers he can select with that picking process?

Input

Input starts with a positive number, the number of test cases C, on a single line ($0 < C \leq 100$). After that, a single line per test case follows describing Sheldon's soldiers on the shelf. The line starts with n ($1 \leq n \leq 2\,000$), the number of toy soldiers, follows by n positive integers s_i ($1 \leq s_i \leq 20\,000$), the height of the soldiers in the order they appear on the shelf.

Output

Output a single line per test case, containing the maximal number m of soldiers he can select, followed by the sum of the heights of the soldiers picked. If there are multiple possibilities to select m soldiers, maximize the sum of their heights.

Sample Input

```
2
5 1 2 3 4 5
6 10 4 11 5 3 4
```

Sample Output

```
1 5
3 20
```

This page is intentionally left (almost) blank.

Problem K

Top Secret II

Once again, things get complicated with Sheldon. He is currently doing research somewhere near the north pole. For an unknown and probably strange reason, he wants to communicate only by means of encrypted messages. To display a message, he uses a bunch of LEDs that he can control remotely. The fact that only the number of illuminated LEDs is important is of course only known to Leonard.
He explained the encryption process to Leonard with simple words:

- define N as the total number of LEDs
- define S as the number of encryption steps
- define X as the number of illuminated LEDs
- a single step is defined as follows: Consider the X illuminated LEDs in turn. For each of these X LEDs either switch on another LED or if no more LED is unlit, switch off all LEDs.

Because Sheldon is able to power/unpower these lights remotely, he is able to send messages to Leonard. Some of the lights are switched on, others are off. Then, he uses a second communication channel (maybe the phone or mail) to communicate the number S of encryption steps to Leonard. Leonard does not know how to decrypt Sheldon's messages. Can you help him?

Input

The first line of the input gives the number of messages ($0 < m < 100$). Each of the messages is provided in a single line with three integers N, S, and X. These numbers are defined as above ($0 < N < 100\,000\,000$, $0 < S < 10\,000$, $0 \leq X \leq N$).

Output

For each test case, print on a single line the number of LEDs that were illuminated before Sheldon started the encryption. If there is no way to determine how many lights were switched on at the beginning, print "`Oops!`" instead of the number.

Sample Input	Sample Output

`4`
`6 2 5`
`22 6 11`
`41 6 11`
`123454 10 1`
	`3`
	`7`
	`Oops!`
	`69564`

This page is intentionally left (almost) blank.

Problem L
Wedding Invitation

Leonard and Sheldon are invited to a wedding. Searching for a proper present they have the idea: Recently, they attended another marriage, where a champagne pyramid was presented. But Leonard and Sheldon aren't the guys who build a regular champagne pyramid.

They form a huge object that consists of N glass canisters of different sizes. The i-th canister has the ground area A_i cm^2. It is placed on top of a box of height b_i cm. The height of the canister itself is h_i cm. Some of these glass canisters are connected by horizontal pipes. Each pipe connects two different canisters at a specific height. Leonard is about to fill in champagne, but he does not want to fill in too much. Therefore, he asks you to calculate what happens if he fills in L milliliters of champagne into the first canister.[3] As you know, Leonard is very accurate and fills in the champagne **very slowly** to keep the champagne from foaming over.

Input

The first line of input is the number of test cases C ($0 < C \leq 100$). The first line of each such test case holds the two integers N and M, the number of canisters ($1 \leq N \leq 20$) and the number of pipes ($0 \leq M \leq 20$). The following N lines contain three integers A_i, b_i and h_i as specified ($1 \leq A_i \leq 100$, $0 \leq b_i \leq 2\,000$, $1 \leq h_i \leq 2\,000$). The next M lines contain integers n_1, n_2 and H ($b_{n_1} < H < b_{n_1} + h_{n_1}$, $b_{n_2} < H < b_{n_2} + h_{n_2}$, $0 \leq n_1 < N$, $0 \leq n_2 < N$) which means that there is a pipe at height H that connects canister n_1 and canister n_2. The next lines gives the number of queries Q ($0 < Q \leq 100$). Each of the following Q lines contains the amount L of champagne in milliliter ($0 \leq L \leq 10\,000$) that Leonard tries to fill into the first canister. You can assume that there are no remains of champagne in the pipes after filling. Furthermore, there are no two pipes at the same height. There is no pair of pipes that connects the same canisters. If a pipe connects can$_a$ to can$_b$, there is no other "path of pipes" from can$_a$ to can$_b$.

Output

For each test case print $Q + 1$ lines. The first line should be "`Case` C`:`", where C is the index of the test case. Afterwards print a line for each query. If there is a canister that overflows, print "`OVERFLOW`" in one line. In case that all L milliliters of champagne remain in the canisters, print N numbers where the i-th number is the amount of milliliters of champagne in the i-th canister with 4 decimal digits after the point. Your output should have an error of at most 10^{-3}.

(Sample Input and Output are provided on the next page.)

[3] Hint: exactly 1 milliliter fits in a canister of volume 1 cm^3

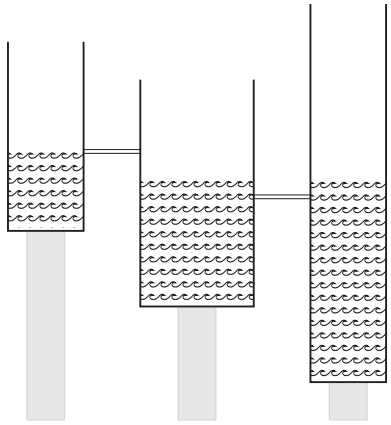


Figure 3 – Illustration of last sample input, filled with 25 milliliters of champagne.

Sample Input

```
4
2 0
1 0 10
100 0 10
2
10
11
2 1
2 2 20
2 4 20
0 1 16
1
40
4 3
24 1073 80
4 625 39
5 643 279
10 552 631
3 2 703
3 0 1091
1 3 636
3
1600
1700
1800
3 2
2 5 5
3 3 6
2 1 10
0 1 7
1 2 6
10
0
1
4
5
13
14
25
41
42
43
```

Sample Output

```
Case 1:
10.0000 0.0000
OVERFLOW
Case 2:
28.0000 12.0000
Case 3:
432.0000 125.1429 0.0000 1042.8571
432.0000 153.7143 0.0000 1114.2857
OVERFLOW
Case 4:
0.0000 0.0000 0.0000
1.0000 0.0000 0.0000
4.0000 0.0000 0.0000
4.0000 1.0000 0.0000
4.0000 9.0000 0.0000
4.0000 9.0000 1.0000
4.0000 10.2000 10.8000
7.7143 17.5714 15.7143
8.0000 18.0000 16.0000
OVERFLOW
```