Problem A
Starship Hakodate-maru
Input: ssh.txt

The surveyor starship Hakodate-maru is famous for her two fuel containers with unbounded capacities. They hold the same type of atomic fuel balls.

There, however, is an inconvenience. The shapes of the fuel containers #1 and #2 are always cubic and regular tetrahedral respectively. Both of the fuel containers should be either empty or filled according to their shapes. Otherwise, the fuel balls become extremely unstable and may explode in the fuel containers. Thus, the number of fuel balls for the container #1 should be a cubic number (n^3 for some $n = 0, 1, 2, 3, \ldots$) and that for the container #2 should be a tetrahedral number ($n(n+1)(n+2)/6$ for some $n = 0, 1, 2, 3, \ldots$).

Hakodate-maru is now at the star base Goryōkaku preparing for the next mission to create a precise and detailed chart of stars and interstellar matters. Both of the fuel containers are now empty. Commander Parus of Goryōkaku will soon send a message to Captain Future of Hakodate-maru on how many fuel balls Goryōkaku can supply. Captain Future should quickly answer to Commander Parus on how many fuel balls she requests before her ship leaves Goryōkaku. Of course, Captain Future and her officers want as many fuel balls as possible.

For example, consider the case Commander Parus offers 151200 fuel balls. If only the fuel container #1 were available (i.e. if the fuel container #2 were unavailable), at most 148877 fuel balls could be put into the fuel container since $148877 = 53 \times 53 \times 53 < 151200 < 54 \times 54 \times 54$. If only the fuel container #2 were available, at most 147440 fuel balls could be put into the fuel container since $147440 = 95 \times 96 \times 97 / 6 < 151200 < 96 \times 97 \times 98 / 6$. Using both of the fuel containers #1 and #2, 151200 fuel balls can be put into the fuel containers since $151200 = 39 \times 39 \times 39 + 81 \times 82 \times 83 / 6$. In this case, Captain Future's answer should be "151200".

Commander Parus's offer cannot be greater than 151200 because of the capacity of the fuel storages of Goryōkaku. Captain Future and her officers know that well.

You are a fuel engineer assigned to Hakodate-maru. Your duty today is to help Captain Future with calculating the number of fuel balls she should request.

Input

The input is a sequence of at most 1024 positive integers. Each line contains a single integer. The sequence is followed by a zero, which indicates the end of data and should not be treated as input. You may assume that none of the input integers is greater than 151200.

Output

The output is composed of lines, each containing a single integer. Each output integer should be the greatest integer that is the sum of a nonnegative cubic number and a nonnegative tetrahedral number and that is not greater than the corresponding input number. No other characters should appear in the output.

Sample Input

```
100
64
50
20
151200
0
```

Output for the Sample Input

```
99
64
47
20
151200
```

Problem B
e-Market
Input: market.txt

The city of Hakodate recently established a commodity exchange market. To participate in the market, each dealer transmits through the Internet an order consisting of his or her name, the type of the order (buy or sell), the name of the commodity, and the quoted price.

In this market a deal can be made only if the price of a sell order is lower than or equal to the price of a buy order. The price of the deal is the mean of the prices of the buy and sell orders, where the mean price is rounded downward to the nearest integer. To exclude dishonest deals, no deal is made between a pair of sell and buy orders from the same dealer. The system of the market maintains the list of orders for which a deal has not been made and processes a new order in the following manner.

- For a new sell order, a deal is made with the buy order with the highest price in the list satisfying the conditions. If there is more than one buy order with the same price, the deal is made with the earliest of them.
- For a new buy order, a deal is made with the sell order with the lowest price in the list satisfying the conditions. If there is more than one sell order with the same price, the deal is made with the earliest of them.

The market opens at 7:00 and closes at 22:00 everyday. When the market closes, all the remaining orders are cancelled. To keep complete record of the market, the system of the market saves all the orders it received everyday.

The manager of the market asked the system administrator to make a program which reports the activity of the market. The report must contain two kinds of information. For each commodity the report must contain information on the lowest, the average and the highest prices of successful deals. For each dealer, the report must contain information on the amounts the dealer paid and received for commodities.

Input

The input contains several data sets. Each data set represents the record of the market on one day. The first line of each data set contains an integer n ($n < 1000$) which is the number of orders in the record. Each line of the record describes an order, consisting of the name of the dealer, the type of the order, the name of the commodity, and the quoted price. They are separated by a single space character.

The name of a dealer consists of capital alphabetical letters and is less than 10 characters in length. The type of an order is indicated by a string, "`BUY`" or "`SELL`". The name of a commodity is a single capital letter. The quoted price is a positive integer less than 1000.

The orders in a record are arranged according to time when they were received and the first line of the record corresponds to the oldest order.

The end of the input is indicated by a line containing a zero.

Output

The output for each data set consists of two parts separated by a line containing two hyphen ('`-`') characters.

The first part is output for commodities. For each commodity, your program should output the name of the commodity and the lowest, the average and the highest prices of successful deals in one line. The name and the prices in a line should be separated by a space character. The average price is rounded downward to the nearest integer. The output should contain only the commodities for which deals are made and the order of the output must be alphabetic.

The second part is output for dealers. For each dealer, your program should output the name of the dealer, the amounts the dealer paid and received for commodities. The name and the numbers in a line should be separated by a space character. The output should contain all the dealers who transmitted orders. The order of dealers in the output must be lexicographic on their names. The lexicographic order is the order in which words in dictionaries are arranged.

The output for each data set should be followed by a line containing ten hyphen ('`-`') characters.

Sample Input

```
3
PERLIS SELL A 300
WILKES BUY A 200
HAMMING SELL A 100
4
BACKUS SELL A 10
FLOYD BUY A 20
IVERSON SELL B 30
BACKUS BUY B 40
7
WILKINSON SELL A 500
MCCARTHY BUY C 300
WILKINSON SELL C 200
DIJKSTRA SELL B 100
BACHMAN BUY A 400
DIJKSTRA BUY A 600
WILKINSON SELL A 300
```

```
2
ABCD SELL X 10
ABC BUY X 15
2
A SELL M 100
A BUY M 100
0
```

Output for the Sample Input

```
A 150 150 150
--
HAMMING 0 150
PERLIS 0 0
WILKES 150 0
----------
A 15 15 15
B 35 35 35
--
BACKUS 35 15
FLOYD 15 0
IVERSON 0 35
----------
A 350 450 550
C 250 250 250
--
BACHMAN 350 0
DIJKSTRA 550 0
MCCARTHY 250 0
WILKINSON 0 1150
----------
X 12 12 12
--
ABC 12 0
ABCD 0 12
----------
--
A 0 0
----------
```

Problem C
Fishnet
Input: fishnet.txt

A fisherman named Etadokah awoke in a very small island. He could see calm, beautiful and blue sea around the island. The previous night he had encountered a terrible storm and had reached this uninhabited island. Some wrecks of his ship were spread around him. He found a square wood-frame and a long thread among the wrecks. He had to survive in this island until someone came and saved him.

In order to catch fish, he began to make a kind of fishnet by cutting the long thread into short threads and fixing them at pegs on the square wood-frame (Figure 1). He wanted to know the sizes of the meshes of the fishnet to see whether he could catch small fish as well as large ones.

The wood-frame is perfectly square with four thin edges one meter long: a bottom edge, a top edge, a left edge, and a right edge. There are n pegs on each edge, and thus there are $4n$ pegs in total. The positions of pegs are represented by their (x, y)-coordinates. Those of an example case with $n = 2$ are depicted in Figures 2 and 3. The position of the ith peg on the bottom edge is represented by $(a_i, 0)$. That on the top edge, on the left edge and on the right edge are represented by $(b_i, 1)$, $(0, c_i)$, and $(1, d_i)$, respectively. The long thread is cut into $2n$ threads with appropriate lengths. The threads are strained between $(a_i, 0)$ and $(b_i, 1)$, and between $(0, c_i)$ and $(1, d_i)$ $(i = 1, \ldots, n)$.

You should write a program that reports the size of the largest mesh among the $(n+1)^2$ meshes of the fishnet made by fixing the threads at the pegs. You may assume that the thread he found is long enough to make the fishnet and that the wood-frame is thin enough for neglecting its thickness.

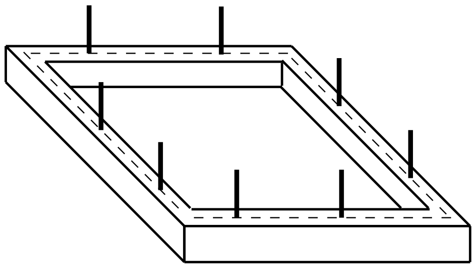
Figure 1. A wood-frame with 8 pegs.

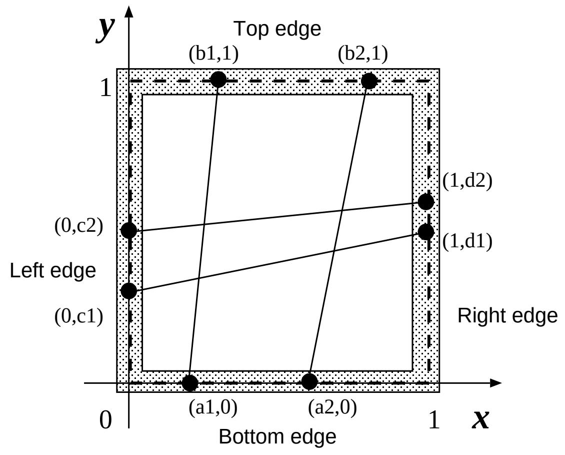

Figure 2. Positions of pegs (indicated by small black circles)

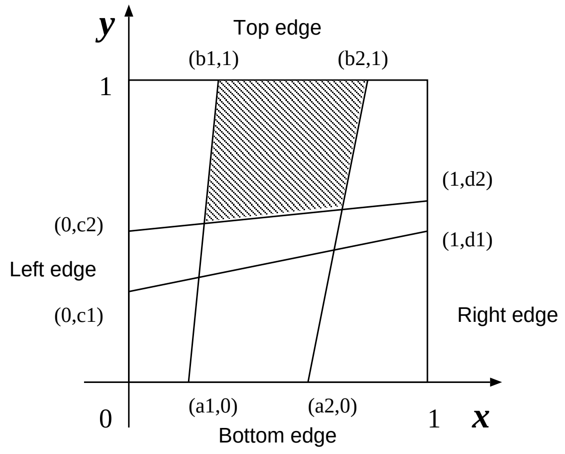

Figure 3. A fishnet and the largest mesh (shaded)

Input

The input consists of multiple subproblems followed by a line containing a zero that indicates the end of input. Each subproblem is given in the following format.

n

$a_1\ a_2 \cdots a_n$

$b_1\ b_2 \cdots b_n$

$c_1\ c_2 \cdots c_n$

$d_1\ d_2 \cdots d_n$

An integer n followed by a newline is the number of pegs on each edge. $a_1, \ldots, a_n$, $b_1, \ldots, b_n$, $c_1, \ldots, c_n$, $d_1, \ldots, d_n$ are decimal fractions, and they are separated by a space character except that a_n, b_n, c_n and d_n are followed by a newline. Each a_i ($i = 1, \ldots, n$) indicates the x-coordinate of the ith peg on the bottom edge. Each b_i ($i = 1, \ldots, n$) indicates the x-coordinate of the ith peg on the top edge. Each c_i ($i = 1, \ldots, n$) indicates the y-coordinate of the ith peg on the left edge. Each d_i ($i = 1, \ldots, n$) indicates the y-coordinate of the ith peg on the right edge. The decimal fractions are represented by 7 digits after the decimal point. In addition you may assume that $0 < n \leq 30$, $0 < a_1 < a_2 < \cdots < a_n < 1$, $0 < b_1 < b_2 < \cdots < b_n < 1$, $0 < c_1 < c_2 < \cdots < c_n < 1$ and $0 < d_1 < d_2 < \cdots < d_n < 1$.

Output

For each subproblem, the size of the largest mesh should be printed followed by a newline. Each value should be represented by 6 digits after the decimal point, and it may not have an error greater than 0.000001.

Sample Input

```
2
0.2000000 0.6000000
0.3000000 0.8000000
0.3000000 0.5000000
0.5000000 0.6000000
2
0.3333330 0.6666670
0.3333330 0.6666670
0.3333330 0.6666670
0.3333330 0.6666670
4
0.2000000 0.4000000 0.6000000 0.8000000
0.1000000 0.5000000 0.6000000 0.9000000
0.2000000 0.4000000 0.6000000 0.8000000
0.1000000 0.5000000 0.6000000 0.9000000
2
0.5138701 0.9476283
0.1717362 0.1757412
0.3086521 0.7022313
0.2264312 0.5345343
1
0.4000000
0.6000000
0.3000000
0.5000000
0
```

Output for the Sample Input

```
0.215657
0.111112
0.078923
0.279223
0.348958
```

Problem D
77377
Input: press.txt

At the risk of its future, International Cellular Phones Corporation (ICPC) invests its resources in developing new mobile phones, which are planned to be equipped with Web browser, mailer, instant messenger, and many other advanced communication tools. Unless members of ICPC can complete this stiff job, it will eventually lose its market share.

You are now requested to help ICPC to develop intriguing text input software for small mobile terminals. As you may know, most phones today have twelve buttons, namely, ten number buttons from "0" to "9" and two special buttons "*" and "#". Although the company is very ambitious, it has decided to follow today's standards and conventions. You should not change the standard button layout, and should also pay attention to the following standard button assignment.

button	letters	button	letters
2	a, b, c	6	m, n, o
3	d, e, f	7	p, q, r, s
4	g, h, i	8	t, u, v
5	j, k, l	9	w, x, y, z

This means that you can only use eight buttons for text input.

Most users of current ICPC phones are rushed enough to grudge wasting time on even a single button press. Your text input software should be economical of users' time so that a single button press is sufficient for each character input. In consequence, for instance, your program should accept a sequence of button presses "77377" and produce the word "press". Similarly, it should translate "77377843288866" into "press the button".

Ummm... It seems impossible to build such text input software since more than one English letter is represented by a digit! For instance, "77377" may represent not only "press" but also any one of 768 ($= 4 \times 4 \times 3 \times 4 \times 4$) character strings. However, we have the good news that the new model of ICPC mobile phones has enough memory to keep a dictionary. You may be able to write a program that filters out *false words*, i.e., strings not listed in the dictionary.

Input

The input consists of multiple data sets, each of which represents a dictionary and a sequence of button presses in the following format.

$$\begin{array}{l} n \\ word_1 \\ \quad\vdots \\ word_n \\ sequence \end{array}$$

n in the first line is a positive integer, representing the number of words in the dictionary. The next n lines, each representing a word in the dictionary, only contain lower case letters from 'a' to 'z'. The order of words in the dictionary is arbitrary (not necessarily in the lexicographic order). No words occur more than once in the dictionary. The last line, *sequence*, is the sequence of button presses, and only contains digits from '2' to '9'.

You may assume that a dictionary has at most one hundred words and that the length of each word is between one and fifty, inclusive. You may also assume that the number of input digits in the *sequence* is between one and three hundred, inclusive.

A line containing a zero indicates the end of the input.

Output

For each data set, your program should print all sequences that can be represented by the input sequence of button presses. Each sequence should be a sequence of words in the dictionary, and should appear in a single line. The order of lines does not matter.

Two adjacent words in a line should be separated by a single space character and the last word should be followed by a single period ('`.`').

Following those output lines, your program should also print a terminating line consisting solely of two hyphens ("`--`"). If there are no corresponding sequences of words, your program should only print the terminating line.

You may assume that for each data set the number of output lines is at most twenty, excluding the terminating line.

Sample Input

```
5
push
press
the
button
bottom
77377843288866
4
i
am
```

```
going
go
42646464
3
a
b
c
333
0
```

Output for the Sample Input

```
press the button.
--
i am going.
i am go go i.
--
--
```

Problem E
Beehives
Input: hive.txt

Taro and Hanako, students majoring in biology, have been engaged long in observations of beehives. Their interest is in finding any egg patterns laid by queen bees of a specific wild species. A queen bee is said to lay a batch of eggs in a short time. Taro and Hanako have never seen queen bees laying eggs. Thus, every time they find beehives, they find eggs just laid in hive cells.

Taro and Hanako have a convention to record an egg layout: they assume the queen bee lays eggs, moving from one cell to an adjacent cell along a path containing no cycles. They record the path of cells with eggs. There is no guarantee in biology for them to find an acyclic path in every case. Yet they have never failed to do so in their observations.

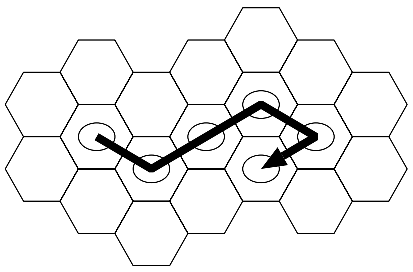
Figure 1.

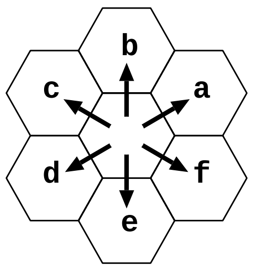

Figure 2.

There are only six possible movements from a cell to an adjacent one, and they agree to write down those six by letters `a`, `b`, `c`, `d`, `e`, and `f` counterclockwise as shown in Figure 2. Thus the layout in Figure 1 may be written down as “`faafd`”.

Taro and Hanako have investigated beehives in a forest independently. Each has his/her own way to approach beehives, protecting oneself from possible bee attacks.

They are asked to report on their work jointly at a conference, and share their own observation records to draft a joint report. At this point they find a serious fault in their convention. They have never discussed which direction, in an absolute sense, should be taken as “`a`”, and thus Figure 2 might be taken as, e.g., Figure 3 or Figure 4. The layout shown in Figure 1 may be recorded differently, depending on the direction looking at the beehive and the path assumed: “`bcbdb`” with combination of Figure 3 and Figure 5, or “`bccac`” with combination of Figure 4 and Figure 6.

A beehive can be observed only from its front side and never from its back, so a layout cannot be confused with its mirror image.

Since they may have observed the same layout independently, they have to find duplicated records in their observations (Of course, they do not record the exact place and the exact time of each observation). Your mission is to help Taro and Hanako by writing a program that checks whether two observation records are made from the same layout.

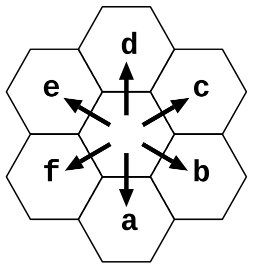

Figure 3.

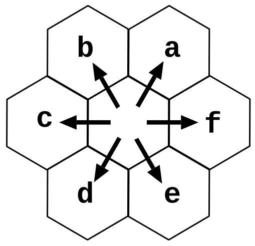

Figure 4.

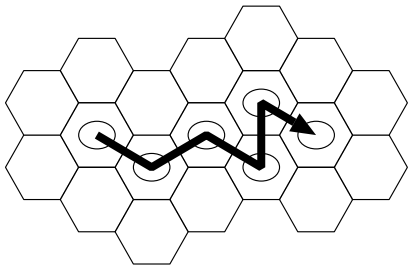

Figure 5.

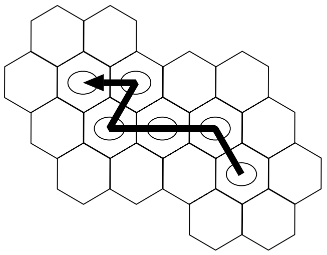

Figure 6.

Input

The input starts with a line containing the number of record pairs that follow. The number is given with at most three digits.

Each record pair consists of two lines of layout records and a line containing a hyphen. Each layout record consists of a sequence of letters `a`, `b`, `c`, `d`, `e`, and `f`. Note that a layout record may be an empty sequence if a queen bee laid only one egg by some reason. You can trust Taro and Hanako in that any of the paths in the input does not force you to visit any cell more than once. Any of lines in the input contain no characters other than those described above, and contain at most one hundred characters.

Output

For each pair of records, produce a line containing either "`true`" or "`false`": "`true`" if the two records represent the same layout, and "`false`" otherwise. A line should not contain any other characters.

Sample Input

```
5
faafd
bcbdb
-
```

```
bcbdb
bccac
-
faafd
aafdd
-
aaafddd
aaaeff
-
aaedd
aafdd
-
```

Output for the Sample Input

```
true
true
false
false
false
```

Problem F
Young, Poor and Busy
Input: young.txt

Ken and Keiko are young, poor and busy. Short explanation: they are students, and ridden with part-time jobs. To make things worse, Ken lives in Hakodate and Keiko in Tokyo. They want to meet, but since they have neither time nor money, they have to go back to their respective jobs immediately after, and must be careful about transportation costs. Help them find the most economical meeting point.

Ken starts from Hakodate, Keiko from Tokyo. They know schedules and fares for all trains, and can choose to meet anywhere including their hometowns, but they cannot leave before 8am and must be back by 6pm in their respective towns. Train changes take no time (one can leave the same minute he/she arrives), but they want to meet for at least 30 minutes in the same city.

There can be up to 100 cities and 2000 direct connections, so you should devise an algorithm clever enough for the task.

Input

The input is a sequence of data sets.

The first line of a data set contains a single integer, the number of connections in the timetable. It is not greater than 2000.

Connections are given one on a line, in the following format.

> *Start_city HH*:*MM Arrival_city HH*:*MM price*

Start_city and *Arrival_city* are composed of up to 16 alphabetical characters, with only the first one in upper case. Departure and arrival times are given in hours and minutes (two digits each, separated by "`:`") from `00:00` to `23:59`. Arrival time is strictly after departure time. The *price* for one connection is an integer between 1 and 10000, inclusive. Fields are separated by spaces.

The end of the input is marked by a line containing a zero.

Output

The output should contain one integer for each data set, the lowest cost possible. This is the total fare of all connections they use.

If there is no solution to a data set, you should output a zero.

The solution to each data set should be given in a separate line.

Sample Input

```
5
Hakodate 08:15 Morioka 12:30 2500
Morioka 14:05 Hakodate 17:30 2500
Morioka 15:30 Hakodate 18:00 3000
Morioka 14:30 Tokyo 17:50 3000
Tokyo 08:30 Morioka 13:35 3000
4
Hakodate 08:15 Morioka 12:30 2500
Morioka 14:04 Hakodate 17:30 2500
Morioka 14:30 Tokyo 17:50 3000
Tokyo 08:30 Morioka 13:35 3000
18
Hakodate 09:55 Akita 10:53 3840
Hakodate 14:14 Akita 16:09 1920
Hakodate 18:36 Akita 19:33 3840
Hakodate 08:00 Morioka 08:53 3550
Hakodate 22:40 Morioka 23:34 3550
Akita 14:23 Tokyo 14:53 2010
Akita 20:36 Tokyo 21:06 2010
Akita 08:20 Hakodate 09:18 3840
Akita 13:56 Hakodate 14:54 3840
Akita 21:37 Hakodate 22:35 3840
Morioka 09:51 Tokyo 10:31 2660
Morioka 14:49 Tokyo 15:29 2660
Morioka 19:42 Tokyo 20:22 2660
Morioka 15:11 Hakodate 16:04 3550
Morioka 23:03 Hakodate 23:56 3550
Tokyo 09:44 Morioka 11:04 1330
Tokyo 21:54 Morioka 22:34 2660
Tokyo 11:34 Akita 12:04 2010
0
```

Output for the Sample Input

```
11000
0
11090
```

Problem G
Nim
Input: nim.txt

Let's play a traditional game Nim. You and I are seated across a table and we have a hundred stones on the table (we know the number of stones exactly). We play in turn and at each turn, you or I can remove one to four stones from the heap. You play first and the one who removed the last stone loses.

In this game, you have a winning strategy. To see this, you first remove four stones and leave 96 stones. No matter how I play, I will end up with leaving 92–95 stones. Then you will in turn leave 91 stones for me (verify this is always possible). This way, you can always leave $5k + 1$ stones for me and finally I get the last stone, sigh. If we initially had 101 stones, on the other hand, I have a winning strategy and you are doomed to lose.

Let's generalize the game a little bit. First, let's make it a team game. Each team has n players and the $2n$ players are seated around the table, with each player having opponents at both sides. Turns round the table so the two teams play alternately. Second, let's vary the maximum number of stones each player can take. That is, each player has his/her own maximum number of stones he/she can take at each turn (The minimum is always one). So the game is asymmetric and may even be unfair.

In general, when played between two teams of experts, the outcome of a game is completely determined by the initial number of stones and the maximum number of stones each player can take at each turn. In other words, either team has a winning strategy.

You are the head-coach of a team. In each game, the umpire shows both teams the initial number of stones and the maximum number of stones each player can take at each turn. Your team plays first. Your job is, given those numbers, to instantaneously judge whether your team has a winning strategy.

Incidentally, there is a rumor that Captain Future and her officers of Hakodate-maru love this game, and they are killing their time playing it during their missions. You wonder where the stones are? Well, they do not have stones but do have plenty of balls in the fuel containers!

Input

The input is a sequence of lines, followed by the last line containing a zero. Each line except the last is a sequence of integers and has the following format.

$$n\ S\ M_1\ M_2\ \cdots\ M_{2n}$$

where n is the number of players in a team, S the initial number of stones, and M_i the maximum number of stones ith player can take. 1st, 3rd, 5th, $\cdots$ players are your team's players and 2nd, 4th, 6th, $\cdots$ the opponents. Numbers are separated by a single space character. You may assume $1 \leq n \leq 10$, $1 \leq M_i \leq 16$, and $1 \leq S < 2^{13}$.

Output

The output should consist of lines each containing either a one, meaning your team has a winning strategy, or a zero otherwise.

Sample Input

```
1 101 4 4
1 100 4 4
3 97 8 7 6 5 4 3
0
```

Output for the Sample Input

```
0
1
1
```

Problem H
Super Star
Input: stars.txt

During a voyage of the starship Hakodate-maru (see Problem A), researchers found strange synchronized movements of stars. Having heard these observations, Dr. Extreme proposed a theory of "super stars". Do not take this term as a description of actors or singers. It is a revolutionary theory in astronomy.

According to this theory, stars we are observing are not independent objects, but only small portions of larger objects called super stars. A super star is filled with invisible (or transparent) material, and only a number of points inside or on its surface shine. These points are observed as stars by us.

In order to verify this theory, Dr. Extreme wants to build motion equations of super stars and to compare the solutions of these equations with observed movements of stars. As the first step, he assumes that a super star is sphere-shaped, and has the smallest possible radius such that the sphere contains all given stars in or on it. This assumption makes it possible to estimate the volume of a super star, and thus its mass (the density of the invisible material is known).

You are asked to help Dr. Extreme by writing a program which, given the locations of a number of stars, finds the smallest sphere containing all of them in or on it. In this computation, you should ignore the sizes of stars. In other words, a star should be regarded as a point. You may assume the universe is a Euclidean space.

Input

The input consists of multiple data sets. Each data set is given in the following format.

n

x_1 y_1 z_1

x_2 y_2 z_2

$\ldots$

x_n y_n z_n

The first line of a data set contains an integer n, which is the number of points. It satisfies the condition $4 \leq n \leq 30$.

The locations of n points are given by three-dimensional orthogonal coordinates: $(x_i,\ y_i,\ z_i)$ $(i = 1, \ldots, n)$. Three coordinates of a point appear in a line, separated by a space character.

Each value is given by a decimal fraction, and is between 0.0 and 100.0 (both ends inclusive). Points are at least 0.01 distant from each other.

The end of the input is indicated by a line containing a zero.

Output

For each data set, the radius of the smallest sphere containing all given points should be printed, each in a separate line. The printed values should have 5 digits after the decimal point. They may not have an error greater than 0.00001.

Sample Input

```
4
10.00000 10.00000 10.00000
20.00000 10.00000 10.00000
20.00000 20.00000 10.00000
10.00000 20.00000 10.00000
4
10.00000 10.00000 10.00000
10.00000 50.00000 50.00000
50.00000 10.00000 50.00000
50.00000 50.00000 10.00000
0
```

Output for the Sample Input

```
7.07107
34.64102
```