Problem Set for JAG Practice Contest 2008

Japanese Alumni Group

Contest Held on 19 Oct 2008

Status of Problems

All problems were newly created by the members of Japanese Alumni Group.

Some portion of the problem statement may modified from the actual practice contest for correction and/or clarification.

The sample input in Problem B was pointed out to contain illegal datasets during the contest. It is corrected in this PDF file.

Problem G is considered not to be properly stated. It might be revised in future.

This problem set was typeset by LaTeX 2_ε with a style file made by a member of JAG from scratch, so that the statement looks like those used in ACM-ICPC Regional Contest held in Japan.

Terms of Use

Problem A
Everlasting...?
Input: A.txt

Everlasting Sa-Ga, a new, hot and very popular role-playing game, is out on October 19, 2008. Fans have been looking forward to a new title of Everlasting Sa-Ga.

Little Jimmy is in trouble. He is a seven-year-old boy, and he obtained the Everlasting Sa-Ga and is attempting to reach the end of the game before his friends. However, he is facing difficulty solving the riddle of the first maze in this game — Everlasting Sa-Ga is notorious in extremely hard riddles like Neverending Fantasy and Forever Quest.

The riddle is as follows. There are two doors on the last floor of the maze: the door to the treasure repository and the gate to the hell. If he wrongly opens the door to the hell, the game is over and his save data will be deleted. Therefore, he should never open the wrong door.

So now, how can he find the door to the next stage? There is a positive integer given for each door — it is a great hint to this riddle. The door to the treasure repository has the integer that gives the larger *key number*. The key number of a positive integer n is the largest prime factor minus the total sum of any other prime factors, where the prime factors are the prime numbers that divide into n without leaving a remainder. Note that each prime factor should be counted only once.

As an example, suppose there are doors with integers 30 and 20 respectively. Since 30 has three prime factors 2, 3 and 5, its key number is $5 - (2 + 3) = 0$. Similarly, since 20 has two prime factors 2 and 5, its key number 20 is $5 - 2 = 3$. Jimmy therefore should open the door with 20.

Your job is to write a program to help Jimmy by solving this riddle.

Input

The input is a sequence of datasets. Each dataset consists of a line that contains two integers a and b separated by a space ($2 \leq a, b \leq 10^6$). It is guaranteed that key numbers of these integers are always different.

The input is terminated by a line with two zeros. This line is not part of any datasets and thus should not be processed.

Output

For each dataset, print in a line '`a`' (without quotes) if the door with the integer a is connected to the treasure repository; print '`b`' otherwise. No extra space or character is allowed.

Sample Input

```
10 15
30 20
0 0
```

Output for the Sample Input

```
a
b
```

Problem B
Headstrong Student
Input: B.txt

You are a teacher at a cram school for elementary school pupils.

One day, you showed your students how to calculate division of fraction in a class of mathematics. Your lesson was kind and fluent, and it seemed everything was going so well — except for one thing. After some experiences, a student Max got so curious about how precise he could compute the quotient. He tried many divisions asking you for a help, and finally found a case where the answer became an infinite fraction. He was fascinated with such a case, so he continued computing the answer. But it was clear for you the answer was an infinite fraction — no matter how many digits he computed, he wouldn't reach the end.

Since you have many other things to tell in today's class, you can't leave this as it is. So you decided to use a computer to calculate the answer in turn of him. Actually you succeeded to persuade him that he was going into a loop, so it was enough for him to know how long he could compute before entering a loop.

Your task now is to write a program which computes where the recurring part starts and the length of the recurring part, for given dividend/divisor pairs. All computation should be done in decimal numbers. If the specified dividend/divisor pair gives a finite fraction, your program should treat the length of the recurring part as 0.

Input

The input consists of multiple datasets. Each line of the input describes a dataset. A dataset consists of two positive integers x and y, which specifies the dividend and the divisor, respectively. You may assume that $1 \leq x < y \leq 1{,}000{,}000$.

The last dataset is followed by a line containing two zeros. This line is not a part of datasets and should not be processed.

Output

For each dataset, your program should output a line containing two integers separated by exactly one blank character.

The former describes the number of digits after the decimal point before the recurring part starts. And the latter describes the length of the recurring part.

Sample Input

```
1 3
1 6
3 5
2 200
25 99
0 0
```

Output for the Sample Input

```
0 1
1 1
1 0
2 0
0 2
```

Problem C
Dig or Climb
Input: C.txt

Benjamin Forest VIII is a king of a country. One of his best friends Nod lives in a village far from his castle. Nod gets seriously sick and is on the verge of death. Benjamin orders his subordinate Red to bring good medicine for him as soon as possible. However, there is no road from the castle to the village. Therefore, Red needs to climb over mountains and across canyons to reach the village. He has decided to get to the village on the shortest path on a map, that is, he will move on the straight line between the castle and the village. Then his way can be considered as polyline with n points $(x_1, y_1) \ldots (x_n, y_n)$ as illustlated in the following figure.

Figure 1: An example route from the castle to the village

Here, x_i indicates the distance between the castle and the point i, as the crow flies, and y_i indicates the height of the point i. The castle is located on the point (x_1, y_1), and the village is located on the point (x_n, y_n).

Red can walk in speed v_w. Also, since he has a skill to cut a tunnel through a mountain horizontally, he can move inside the mountain in speed v_c.

Your job is to write a program to the minimum time to get to the village.

Input

The input is a sequence of datasets. Each dataset is given in the following format:

n

v_w v_c
x_1 y_1
...
x_n y_n

You may assume all the following: $n \leq 1{,}000$, $1 \leq v_w, v_c \leq 10$, $-10{,}000 \leq x_i, y_i \leq 10{,}000$, and $x_i < x_j$ for all $i < j$.

The input is terminated in case of $n = 0$. This is not part of any datasets and thus should not be processed.

Output

For each dataset, you should print the minimum time required to get to the village in a line. Each minimum time should be given as a decimal with an arbitrary number of fractional digits and with an absolute error of at most 10^{-6}. No extra space or character is allowed.

Sample Input

```
3
2 1
0 0
50 50
100 0
3
1 1
0 0
50 50
100 0
3
1 2
0 0
50 50
100 0
3
2 1
0 0
100 100
150 50
6
1 2
0 0
50 50
100 0
150 0
```

```
200 50
250 0
0
```

Output for the Sample Input

```
70.710678
100.000000
50.000000
106.066017
150.000000
```

Problem D
Rotation Estimation
Input: D.txt

Mr. Nod is an astrologist and has defined a new constellation. He took two photos of the constellation to foretell a future of his friend. The constellation consists of n stars. The shape of the constellation in these photos are the same, but the angle of them are different because these photos were taken on a different day. He foretells a future by the difference of the angle of them.

Your job is to write a program to calculate the difference of the angle of two constellation.

Input

The input is a sequence of datasets. Each dataset is given in the following format:

n
$x_{1,1}$ $y_{1,1}$
...
$x_{1,n}$ $y_{1,n}$
$x_{2,1}$ $y_{2,1}$
...
$x_{2,n}$ $y_{2,n}$

The first line of each dataset contains a positive integers n ($n \leq 1{,}000$). The next n lines contain two real numbers $x_{1,i}$ and $y_{1,i}$ ($|x_{1,i}|, |y_{1,i}| \leq 100$), where ($x_{1,i}$, $y_{1,i}$) denotes the coordinates of the i-th star of the constellation in the first photo. The next n lines contain two real numbers $x_{2,i}$ and $y_{2,i}$ ($|x_{2,i}|, |y_{2,i}| \leq 100$), where ($x_{2,i}$, $y_{2,i}$) denotes the coordinates of the i-th star of the constellation in the second photo.

Note that the ordering of the stars does *not* matter for the sameness. It is guaranteed that distance between every pair of stars in each photo is larger than 10^{-5}.

The input is terminated in case of $n = 0$. This is not part of any datasets and thus should not be processed.

Output

For each dataset, you should print a non-negative real number which is the difference of the angle of the constellation in the first photo and in the second photo. The difference should be

in radian, and should not be negative. If there are two or more solutions, you should print the smallest one. The difference may be printed with any number of digits after decimal point, provided the absolute error does not exceed 10^{-7}. No extra space or character is allowed.

Sample Input

```
3
0.0 0.0
1.0 1.0
0.0 1.0
3.0 3.0
2.0 2.0
3.0 2.0
0
```

Output for the Sample Input

```
3.14159265359
```

Problem E
Optimal Rest
Input: E.txt

Music Macro Language (MML) is a language for textual representation of musical scores. Although there are various dialects of MML, all of them provide a set of commands to describe scores, such as commands for notes, rests, octaves, volumes, and so forth.

In this problem, we focus on rests, i.e. intervals of silence. Each rest command consists of a command specifier '`R`' followed by a duration specifier. Each duration specifier is basically one of the following numbers: '`1`', '`2`', '`4`', '`8`', '`16`', '`32`', and '`64`', where '`1`' denotes a whole (1), '`2`' a half (1/2), '`4`' a quarter (1/4), '`8`' an eighth (1/8), and so on. This number is called the *base duration*, and optionally followed by one or more dots. The first dot adds the duration by the half of the base duration. For example, '`4.`' denotes the duration of '`4`' (a quarter) plus '`8`' (an eighth, i.e. the half of a quarter), or simply 1.5 times as long as '`4`'. In other words, '`R4.`' is equivalent to '`R4R8`'. In case with two or more dots, each extra dot extends the duration by the half of the previous one. Thus '`4..`' denotes the duration of '`4`' plus '`8`' plus '`16`', '`4...`' denotes the duration of '`4`' plus '`8`' plus '`16`' plus '`32`', and so on. The duration extended by dots cannot be shorter than '`64`'. For exapmle, neither '`64.`' nor '`16...`' will be accepted since both of the last dots indicate the half of '`64`' (i.e. the duration of 1/128).

In this problem, you are required to write a program that finds the shortest expressions equivalent to given sequences of rest commands.

Input

The input consists of multiple datasets. The first line of the input contains the number of datasets N. Then, N datasets follow, each containing a sequence of valid rest commands in one line. You may assume that no sequence contains more than 100,000 characters.

Output

For each dataset, your program should output the shortest expression in one line. If there are multiple expressions of the shortest length, output the lexicographically smallest one.

Sample Input

```
3
R2R2
R1R2R4R8R16R32R64
```

```
R1R4R16
```

Output for the Sample Input

```
R1
R1......
R16R1R4
```

Problem F
Controlled Tournament
Input: F.txt

National Association of Tennis is planning to hold a tennis competition among professional players. The competition is going to be a knockout tournament, and you are assigned the task to make the arrangement of players in the tournament.

You are given the detailed report about all participants of the competition. The report contains the results of recent matches between all pairs of the participants. Examining the data, you've noticed that it is only up to the opponent whether one player wins or not.

Since one of your special friends are attending the competition, you want him to get the best prize. So you want to know the possibility where he wins the gold medal. However it is not so easy to figure out because there are many participants. You have decided to write a program which calculates the number of possible arrangements of tournament in which your friend wins the gold medal.

In order to make your trick hidden from everyone, you need to avoid making a factitive tournament tree. So you have to minimize the height of your tournament tree.

Input

The input consists of multiple datasets. Each dataset has the format as described below.

N M
R_{11} R_{12} ... R_{1N}
R_{21} R_{22} ... R_{2N}
...
R_{N1} R_{N2} ... R_{NN}

N ($2 \leq N \leq 16$) is the number of player, and M ($1 \leq M \leq N$) is your friend's ID (numbered from 1). R_{ij} is the result of a match between the i-th player and the j-th player. When i-th player always wins, $R_{ij} = 1$. Otherwise, $R_{ij} = 0$. It is guaranteed that the matrix is consistent: for all $i \neq j$, $R_{ij} = 0$ if and only if $R_{ji} = 1$. The diagonal elements R_{ii} are just given for convenience and are always 0.

The end of input is indicated by a line containing two zeros. This line is not a part of any datasets and should not be processed.

Output

For each dataset, your program should output in a line the number of possible tournaments in which your friend wins the first prize.

Sample Input

```
2 1
0 1
0 0
2 1
0 0
1 0
3 3
0 1 1
0 0 1
0 0 0
3 3
0 1 0
0 0 0
1 1 0
3 1
0 1 0
0 0 0
1 1 0
3 3
0 1 0
0 0 1
1 0 0
6 4
0 0 0 0 0 1
1 0 1 0 1 0
1 0 0 1 1 0
1 1 0 0 1 0
1 0 0 0 0 0
0 1 1 1 1 0
7 2
0 1 0 0 0 1 0
0 0 1 0 1 1 1
1 0 0 1 1 0 0
1 1 0 0 0 1 0
1 0 0 1 0 0 1
0 0 1 0 1 0 0
1 0 1 1 0 1 0
8 6
```

```
0 0 0 0 1 0 0 0
1 0 1 1 0 0 0 0
1 0 0 0 1 0 0 0
1 0 1 0 0 1 0 1
0 1 0 1 0 0 1 0
1 1 1 0 1 0 0 1
1 1 1 1 0 1 0 0
1 1 1 0 1 0 1 0
0 0
```

Output for the Sample Input

```
1
0
0
3
0
1
11
139
78
```

Problem G
Entangled Tree
Input: G.txt

The electronics division in Ishimatsu Company consists of various development departments for electronic devices including disks and storages, network devices, mobile phones, and many others. Each department covers a wide range of products. For example, the department of disks and storages develops internal and external hard disk drives, USB thumb drives, solid-state drives, and so on. This situation brings staff in the product management division difficulty categorizing these numerous products because of their poor understanding of computer devices.

One day, a staff member suggested a tree-based diagram named a *category diagram* in order to make their tasks easier. A category diagram is depicted as follows. Firstly, they prepare one large sheet of paper. Secondly, they write down the names of the development departments on the upper side of the sheet. These names represent the *start nodes* of the diagram. Each start node is connected to either a single split node or a single end node (these nodes will be mentioned soon later). Then they write down a number of questions that distinguish features of products in the middle, and these questions represent the *split nodes* of the diagram. Each split node is connected with other split nodes and end nodes, and each line from a split node is labeled with the answer to the question. Finally, they write down all category names on the lower side, which represents the *end nodes*.

The classification of each product is done like the following. They begin with the start node that corresponds to the department developing the product. Visiting some split nodes, they traces the lines down until they reach one of the end nodes labeled with a category name. Then they find the product classified into the resultant category.

The visual appearance of a category diagram makes the diagram quite understandable even for non-geek persons. However, product managers are not good at drawing the figures by hand, so most of the diagrams were often messy due to many line crossings. For this reason, they hired you, a talented programmer, to obtain the *clean diagrams* equivalent to their diagrams. Here, we mean the clean diagrams as those with no line crossings.

Your task is to write a program that finds the clean diagrams. For simplicity, we simply ignore the questions of the split nodes, and use integers from 1 to N instead of the category names.

Input

The input consists of multiple datasets. Each dataset follows the format below:

$N\ M\ Q$

$split\_node\_info_1$
$split\_node\_info_2$
...
$split\_node\_info_M$
$query_1$
$query_2$
...
$query_Q$

The first line of each dataset contains three integers N ($1 \leq N \leq 100000$), M ($0 \leq M \leq N-1$), and Q ($1 \leq Q \leq 1000$, $Q \leq N$), representing the number of end nodes and split nodes, and the number of queries respectively. Then M lines describing the split nodes follow. Each split node is described in the format below:

$Y\ L\ label_1\ label_2\ \ldots$

The first two integers, Y ($0 \leq Y \leq 10^9$) and L, which indicates the y-coordinate where the split node locates (the smaller is the higher) and the size of a label list. After that, L integer numbers of end node labels which directly or indirectly connected to the split node follow. This is a key information for node connections. A split node A is connected to another split node B if and only if both A and B refer (at least) one identical end node in their label lists, and the y-coordinate of B is the lowest of all split nodes referring identical end nodes and located below A. The split node is connected to the end node if and only if that is the lowest node among all nodes which contain the same label as the end node's label. The start node is directly connected to the end node, if and only if the end node is connected to none of the split nodes.

After the information of the category diagram, Q lines of integers follow. These integers indicate the horizontal positions of the end nodes in the diagram. The leftmost position is numbered 1.

The input is terminated by the dataset with $N = M = Q = 0$, and this dataset should not be processed.

Output

Your program must print the Q lines, each of which denotes the label of the end node at the position indicated by the queries in the clean diagram. One blank line must follow after the output for each dataset.

Sample Input

```
3 2 3
10 2 1 2
20 2 3 2
1
```

```
2
3
5 2 5
10 3 1 2 4
20 3 2 3 5
1
2
3
4
5
4 1 4
10 2 1 4
1
2
3
4
4 3 4
30 2 1 4
20 2 2 4
10 2 3 4
1
2
3
4
4 3 4
10 2 1 4
20 2 2 4
30 2 3 4
1
2
3
4
4 3 4
10 2 1 2
15 2 1 4
20 2 2 3
1
2
3
4
3 2 3
10 2 2 3
20 2 1 2
1
2
```

```
3
1 0 1
1
0 0 0
```

Output for the Sample Input

```
1
2
3

1
2
3
5
4

1
4
2
3

1
4
2
3

1
2
3
4

1
4
2
3

1
2
3

1
```

Problem H
Ramen Shop
Input: H.txt

Ron is a master of a ramen shop.

Recently, he has noticed some customers wait for a long time. This has been caused by lack of seats during lunch time. Customers loses their satisfaction if they waits for a long time, and even some of them will give up waiting and go away. For this reason, he has decided to increase seats in his shop. To determine how many seats are appropriate, he has asked you, an excellent programmer, to write a simulator of customer behavior.

Customers come to his shop in groups, each of which is associated with the following four parameters:

- T_i: when the group comes to the shop
- P_i: number of customers
- W_i: how long the group can wait for their seats
- E_i: how long the group takes for eating

The i-th group comes to the shop with P_i customers together at the time T_i. If P_i successive seats are available at that time, the group takes their seats immediately. Otherwise, they waits for such seats being available. When the group fails to take their seats within the time W_i (inclusive) from their coming and strictly before the closing time, they give up waiting and go away. In addition, if there are other groups waiting, the new group cannot take their seats until the earlier groups are taking seats or going away.

The shop has N counters numbered uniquely from 1 to N. The i-th counter has C_i seats. The group prefers "seats with a greater distance to the nearest group." Precisely, the group takes their seats according to the criteria listed below. Here, S_L denotes the number of successive empty seats on the left side of the group after their seating, and S_R the number on the right side. S_L and S_R are considered to be infinity if there are no other customers on the left side and on the right side respectively.

1. Prefers seats maximizing $\min\{S_L, S_R\}$.
2. If there are multiple alternatives meeting the first criterion, prefers seats maximizing $\max\{S_L, S_R\}$.

3. If there are still multiple alternatives, prefers the counter of the smallest number.
4. If there are still multiple alternatives, prefers the leftmost seats.

When multiple groups are leaving the shop at the same time and some other group is waiting for available seats, seat assignment for the waiting group should be made after all the finished groups leave the shop.

Your task is to calculate the average satisfaction over customers. The satisfaction of a customer in the i-th group is given as follows:

- If the group goes away without eating, -1.
- Otherwise, $(W_i - t_i)/W_i$ where t_i is the actual waiting time for the i-th group (the value ranges between 0 to 1 inclusive).

Input

The input consists of multiple datasets.

Each dataset has the following format:

$N\ M\ T$
$C_1\ C_2\ \cdots\ C_N$
$T_1\ P_1\ W_1\ E_1$
$T_2\ P_2\ W_2\ E_2$
$\cdots$
$T_M\ P_M\ W_M\ E_M$

N indicates the number of counters, M indicates the number of groups and T indicates the closing time. The shop always opens at the time 0. All input values are integers.

You can assume that $1 \leq N \leq 100$, $1 \leq M \leq 10000$, $1 \leq T \leq 10^9$, $1 \leq C_i \leq 100$, $0 \leq T_1 < T_2 < \cdots < T_M < T$, $1 \leq P_i \leq \max C_i$, $1 \leq W_i \leq 10^9$ and $1 \leq E_i \leq 10^9$.

The input is terminated by a line with three zeros. This is not part of any datasets.

Output

For each dataset, output the average satisfaction over all customers in a line. Each output value may be printed with an arbitrary number of fractional digits, but may not contain an absolute error greater than 10^{-9}.

Sample Input

```
1 4 100
7
10 1 50 50
15 2 50 50
25 1 50 50
35 3 50 50
1 2 100
5
30 3 20 50
40 4 40 50
1 2 100
5
49 3 20 50
60 4 50 30
1 2 100
5
50 3 20 50
60 4 50 30
2 3 100
4 2
10 4 20 20
30 2 20 20
40 4 20 20
0 0 0
```

Output for the Sample Input

```
0.7428571428571429
0.4285714285714285
0.5542857142857143
-0.1428571428571428
0.8000000000000000
```

Problem I
Cousin's Aunt
Input: I.txt

Sarah is a girl who likes reading books.

One day, she wondered about the relationship of a family in a mystery novel. The story said,

- B is A's father's brother's son, and
- C is B's aunt.

Then she asked herself, "So how many degrees of kinship are there between A and C?"

There are two possible relationships between B and C, that is, C is either B's father's sister or B's mother's sister in the story. If C is B's father's sister, C is in the third degree of kinship to A (A's father's sister). On the other hand, if C is B's mother's sister, C is in the fifth degree of kinship to A (A's father's brother's wife's sister).

You are a friend of Sarah's and good at programming. You can help her by writing a general program to calculate the maximum and minimum degrees of kinship between A and C under given relationship.

The relationship of A and C is represented by a sequence of the following basic relations: father, mother, son, daughter, husband, wife, brother, sister, grandfather, grandmother, grandson, granddaughter, uncle, aunt, nephew, and niece. Here are some descriptions about these relations:

- X's brother is equivalent to X's father's or mother's son not identical to X.
- X's grandfather is equivalent to X's father's or mother's father.
- X's grandson is equivalent to X's son's or daughter's son.
- X's uncle is equivalent to X's father's or mother's brother.
- X's nephew is equivalent to X's brother's or sister's son.
- Similar rules apply to sister, grandmother, granddaughter, aunt and niece.

In this problem, you can assume there are none of the following relations in the family: adoptions, marriages between relatives (i.e. the family tree has no cycles), divorces, remarriages, bigamous marriages and same-sex marriages.

The degree of kinship is defined as follows:

- The distance from X to X's father, X's mother, X's son or X's daughter is one.
- The distance from X to X's husband or X's wife is zero.
- The degree of kinship between X and Y is equal to the shortest distance from X to Y deduced from the above rules.

Input

The input contains multiple datasets. The first line consists of a positive integer that indicates the number of datasets.

Each dataset is given by one line in the following format:

`C is A('s` *relation*)*

Here, *relation* is one of the following: `father`, `mother`, `son`, `daughter`, `husband`, `wife`, `brother`, `sister`, `grandfather`, `grandmother`, `grandson`, `granddaughter`, `uncle`, `aunt`, `nephew`, `niece`. An asterisk denotes zero or more occurance of portion surrounded by the parentheses. The number of relations in each dataset is at most ten.

Output

For each dataset, print a line containing the maximum and minimum degrees of kinship separated by exact one space. No extra characters are allowed of the output.

Sample Input

```
7
C is A's father's brother's son's aunt
C is A's mother's brother's son's aunt
C is A's son's mother's mother's son
C is A's aunt's niece's aunt's niece
C is A's father's son's brother
C is A's son's son's mother
C is A
```

Output for the Sample Input

```
5 3
5 1
2 2
```

```
6 0
2 0
1 1
0 0
```

Problem J
Colony Maintenance
Input: J.txt

It is the year of 2xxx. Human beings have migrated out of the Earth and resided in space colonies. Because of the severe environment of the space, which comes from cosmic rays and meteorites, those colonies have to be continuously repaired by maintenance robots. In this problem, you are requested to write a program that calculates the shortest distance for a robot to move from the present point to the next repair point on a colony.

Each colony can be modeled as a polycube, that is, one or more cubes of the same size all joined by their faces. Note that, as forming a polycube, all cubes of each colony are connected. This implies every cube has at least one face coincident with a face of another cube, except for colonies with a single cube. The figure below illustrates a couple of example colonies.

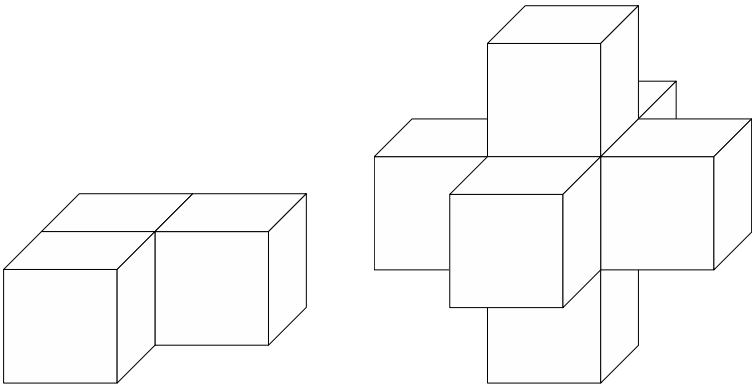

Figure 2: Example colonies

The maintenance robot can move only on the surface of the polycubes, that is, on faces not in common with other cubes. In addition, due to the structure of the colonies, move of the robot beyond a face is restricted to the following cases: (a) the robot is moving between adjacent faces of the same cube; (b) the robot is moving between adjacent faces of adjacent cubes; and (c) the robot is moving along the inner side of an L-shape, namely, the robot is moving between adjacent faces of two cubes that have the common adjacent cube. Here, adjacent faces denote those with the common edge, and adjacent cubes denote those with the common face.

For the purpose of this problem, we consider an xyz-space (i.e. a three-dimensional space with the Cartesian coordinate system) where all cubes are placed in such a way that each edge is parallel to x-axis, y-axis, or z-axis. Also, the edge length of the cubes is scaled to 100.

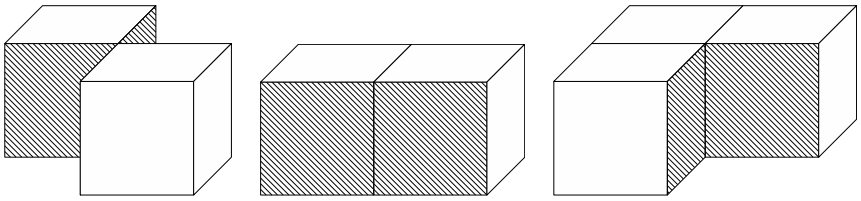

Figure 3: Possible moves

Input

The input is a sequence of datasets, each of which has the following format:

n
$x_1\ y_1\ z_1$
...
$x_n\ y_n\ z_n$
$sx\ sy\ sz\ dx\ dy\ dz$

n is the number of cubes of a colony ($1 \leq n \leq 16$). (x_i, y_i, z_i) represents the coordinates of the center of the i-th cube, where each coordinate value is guaranteed to be a multiple of 100 (i.e. the edge length of the cubes). (sx, sy, sz) and (dx, dy, dz) represent the robot's present point and the next repair point respectively. You may assume that these two points are always different and that neither of them lies on any edge. All coordinate values are integers between -2000 and 2000 inclusive.

The input is terminated by a line with a single zero, which is not part of any datasets and hence should not be processed.

Output

For each dataset, print the distance of the shortest route from the present point to the next repair point. The distance may be printed with any number of digits after the decimal point, provided the absolute error does not exceed 10^{-8}.

Sample Input

```
1
0 0 0
0 0 50 30 40 50
1
0 0 0
50 0 0 0 50 0
```

```
2
0 0 0
100 0 0
0 0 50 100 0 50
3
0 0 0
100 0 0
0 100 0
100 50 0 50 100 0
7
0 0 0
100 0 0
-100 0 0
0 100 0
0 -100 0
0 0 100
0 0 -100
150 0 0 -150 0 0
6
0 0 0
100 0 0
0 100 0
100 100 0
0 0 100
100 100 100
100 0 50 0 100 50
6
0 0 0
100 0 0
0 100 0
100 100 0
0 0 100
100 100 100
100 100 150 0 0 150
6
0 0 0
100 0 0
0 100 0
100 100 100
100 0 100
0 100 100
0 0 50 100 100 50
8
0 0 0
-100 0 0
```

```
-100 100 0
-100 200 0
0 200 0
100 200 0
100 200 100
100 100 100
0 0 50 100 100 50
0
```

Output for the Sample Input

```
50.0
100.0
100.0
100.0
416.2277660168
141.4213562373
316.2277660168
341.4213562373
341.4213562373
```

The figure below shows how the robot will move in the sixth case:

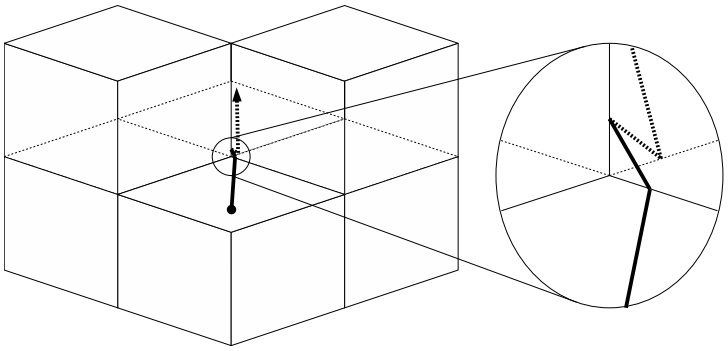

Figure 4: The robot's move in the 6th case