Problem Set for Practice Contest

Japanese Alumni Group

Contest Held on: 22 Oct 2006

This problem set contains nine problems identified by A through I.

All problems were newly created by the members of Japanese Alumni Group. You may use the problems in any form, entirely or in part, with or without modification, for any purposes, without prior or posterior consent to Japanese Alumni Group, provided that your use is made solely at your own risk.

Problem B is based on the game RPS-15 created by David C. Lovelace, which is open to the public at `http://www.umop.com/rps15.htm`. Japanese Alumni Group has no relationship with Mr. Lovelace.

Note: The sample input for Problem H was found to be incorrect by a clarification from a team in the contest proper. It is corrected in this file.

Problem A
Ruins
Input: A.txt

In 1936, a dictator Hiedler who aimed at world domination had a deep obsession with *the Lost Ark*. A person with this ark would gain mystic power according to legend. To break the ambition of the dictator, ACM (the Alliance of Crusaders against Mazis) entrusted a secret task to an archeologist Indiana Johns. Indiana stepped into a vast and harsh desert to find the ark.

Indiana finally found an underground treasure house at a ruin in the desert. The treasure house seems storing the ark. However, the door to the treasure house has a special lock, and it is not easy to open the door.

To open the door, he should solve a problem raised by two positive integers a and b inscribed on the door. The problem requires him to find the minimum sum of squares of differences for all pairs of two integers adjacent in a sorted sequence that contains four positive integers a_1, a_2, b_1 and b_2 such that $a = a_1 a_2$ and $b = b_1 b_2$. Note that these four integers does *not* need to be different. For instance, suppose 33 and 40 are inscribed on the door, he would have a sequence $\langle 3, 5, 8, 11 \rangle$ as 33 and 40 can be decomposed into 3×11 and 5×8 respectively. This sequence gives the sum $(5-3)^2 + (8-5)^2 + (11-8)^2 = 22$, which is the smallest sum among all possible sorted sequences. This example is included as the first data set in the sample input.

Once Indiana fails to give the correct solution, he will suffer a calamity by a curse. On the other hand, he needs to solve the problem as soon as possible, since many pawns under Hiedler are also searching for the ark, and they might find the same treasure house. He will be immediately killed if this situation happens. So he decided to solve the problem by your computer program.

Your task is to write a program to solve the problem presented above.

Input

The input consists of a series of data sets. Each data set is given by a line that contains two positive integers not greater than 10,000.

The end of the input is represented by a pair of zeros, which should not be processed.

Output

For each data set, print a single integer that represents the minimum square sum in a line. No extra space or text should appear.

Sample Input

```
33 40
57 144
```

```
0 0
```

Output for the Sample Input

```
22
94
```

Problem B
Hyper Rock-Scissors-Paper
Input: B.txt

Rock-Scissors-Paper is a game played with hands and often used for random choice of a person for some purpose. Today, we have got an extended version, namely, *Hyper Rock-Scissors-Paper* (or Hyper RSP for short).

In a game of Hyper RSP, the players simultaneously presents their hands forming any one of the following 15 gestures: Rock, Fire, Scissors, Snake, Human, Tree, Wolf, Sponge, Paper, Air, Water, Dragon, Devil, Lightning, and Gun.

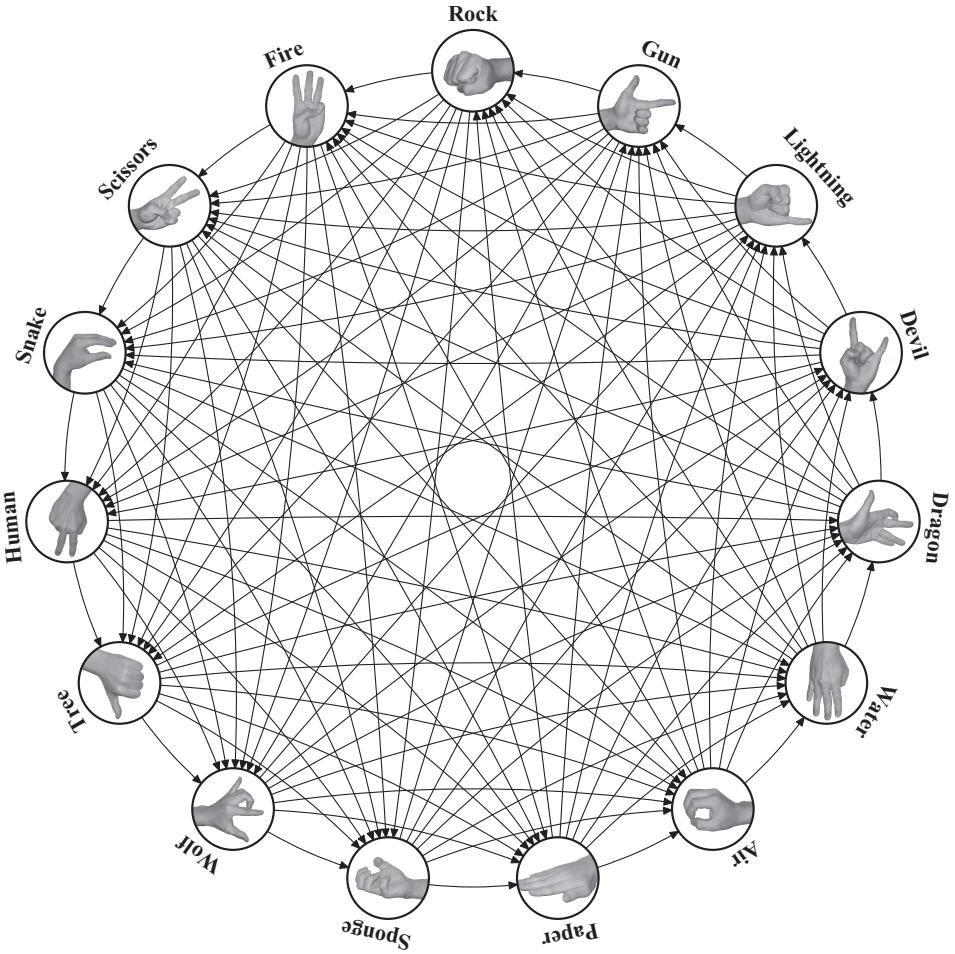

Figure 1: Hyper Rock-Scissors-Paper

The arrows in the figure above show the defeating relation. For example, Rock defeats Fire, Scissors, Snake, Human, Tree, Wolf, and Sponge. Fire defeats Scissors, Snake, Human, Tree, Wolf, Sponge, and Paper. Generally speaking, each hand defeats other seven hands located after in anti-clockwise order in the figure. A player is said to *win* the game if the player's hand defeats at least one of the other hands, *and* is not defeated by any of the other hands.

Your task is to determine the winning hand, given multiple hands presented by the players.

Input

The input consists of a series of data sets. The first line of each data set is the number N of the players ($N < 1000$). The next N lines are the hands presented by the players.

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

Output

For each data set, output the winning hand in a single line. When there are no winners in the game, output "`Draw`" (without quotes).

Sample Input

```
8
Lightning
Gun
Paper
Sponge
Water
Dragon
Devil
Air
3
Rock
Scissors
Paper
0
```

Output for the Sample Input

```
Sponge
Draw
```

Problem C
Online Quiz System
Input: C.txt

ICPC (Internet Contents Providing Company) is working on a killer game named Quiz Millionaire Attack. It is a quiz system played over the Internet. You are joining ICPC as an engineer, and you are responsible for designing a protocol between clients and the game server for this system. As bandwidth assigned for the server is quite limited, data size exchanged between clients and the server should be reduced as much as possible. In addition, all clients should be well synchronized during the quiz session for a simultaneous play. In particular, much attention should be paid to the delay on receiving packets.

To verify that your protocol meets the above demands, you have decided to simulate the communication between clients and the server and calculate the data size exchanged during one game.

A game has the following process. First, players participating and problems to be used are fixed. All players are using the same client program and already have the problem statements downloaded, so you don't need to simulate this part. Then the game begins. The first problem is presented to the players, and the players answer it within a fixed amount of time. After that, the second problem is presented, and so forth. When all problems have been completed, the game ends. During each problem phase, the players are notified of what other players have done. Before you submit your answer, you can know who have already submitted their answers. After you have submitted your answer, you can know what answers are submitted by other players.

When each problem phase starts, the server sends a *synchronization packet for problem-start* to all the players, and begins a polling process. Every 1,000 milliseconds after the beginning of polling, the server checks whether it received new answers from the players strictly before that moment, and if there are any, sends a notification to all the players:

- If a player hasn't submitted an answer yet, the server sends it a *notification packet type A describing others' answers* about the newly received answers.
- If a player is one of those who submitted the newly received answers, the server sends it a *notification packet type B describing others' answers* about all the answers submitted by other players (i.e. excluding the player him/herself's answer) strictly before that moment.
- If a player has already submitted an answer, the server sends it a *notification packet type B describing others' answers* about the newly received answers.
- Note that, in all above cases, notification packets (both types A and B) must contain information about at least one player, and otherwise a notification packet will not be sent.

When 20,000 milliseconds have passed after sending the *synchronization packet for problem-start*, the server sends notification packets of type A or B if needed, and then sends a *synchronization packet for problem-end* to all the players, to terminate the problem.

On the other hand, players will be able to answer the problem after receiving the *synchronization packet for problem-start* and before receiving the *synchronization packet for problem-end*. Answers will be sent using an *answer packet*.

The packets referred above have the formats shown by the following tables.

Content	**Size**
packet header	3

Table 1: Syncronization packet for problem-start

Content	**Size**
packet header	3
player ID	1
answer data size ($= L$)	1
answer data	L

Table 2: Answer packet

Content	**Size**
packet header	3
number of other players' answers ($= N$)	1
(*the following repeated N times*)	
player ID	1

Table 3: Notification packet type A describing others' answers

Content	**Size**
packet header	3
number of other players' answers ($= N$)	1
(*the following repeated N times*)	
player ID	1
answer data size ($= L_i$)	1
answer data	L_i

Table 4: Notification packet type B describing others' answers

Content	**Size**
packet header	3
result	1

Table 5: Syncronization packet for problem-end

Input

The input consists of multiple test cases. Each test case begins with a line consisting of two integers M and N ($1 \leq M, N \leq 100$), denoting the number of players and problems, respectively. The next line contains M non-negative integers D_0, D_1, ... , D_{M-1}, denoting the communication delay between each players and the server (players are assigned ID's ranging from 0 to $M - 1$, inclusive). Then follow N blocks describing the submissions for each problem. Each block begins with a line containing an integer L, denoting the number of players that submitted an answer for that problem. Each of the following L lines gives the answer from one player, described by three fields P, T, and A separated by a whitespace. Here P is an integer denoting the player ID, T is an integer denoting the time elapsed from

the reception of *synchronization packet for problem-start* and the submission on the player's side, and A is an alphanumeric string denoting the player's answer, whose length is between 1 and 9, inclusive. Those L lines may be given in an arbitrary order. You may assume that all *answer packets* will be received by the server within 19,999 milliseconds (inclusive) after sending *synchronization packet for problem-start*.

The input is terminated by a line containing two zeros.

Output

For each test case, you are to print $M + 1$ lines. In the first line, print the data size sent and received by the server, separated by a whitespace. In the next M lines, print the data size sent and received by each player, separated by a whitespace, in ascending order of player ID. Print a blank line between two consecutive test cases.

Sample Input

```
3 2
1 2 10
3
0 3420 o
1 4589 o
2 4638 x
3
1 6577 SUZUMIYA
2 7644 SUZUMIYA
0 19979 YASUZUMI
4 2
150 150 150 150
4
0 1344 HOGEHOGE
1 1466 HOGEHOGE
2 1789 HOGEHOGE
3 19100 GEHO
2
2 1200 SETTEN
3 700 SETTEN
0 0
```

Output for the Sample Input

```
177 57
19 58
19 57
19 62

253 70
13 58
13 58
24 66
20 71
```

Problem D
Rock Man
Input: D.txt

You were the master craftsman in the stone age, and you devoted your life to carving many varieties of tools out of natural stones. Your works have ever been displayed respectfully in many museums, even in 2006 A.D. However, most people visiting the museums do not spend so much time to look at your works. Seeing the situation from the Heaven, you brought back memories of those days you had frantically lived.

One day in your busiest days, you got a number of orders for making tools. Each order requested one tool which was different from the other orders. It often took some days to fill one order without special tools. But you could use tools which you had already made completely, in order to carve new tools more quickly. For each tool in the list of the orders, there was exactly one tool which could support carving it.

In those days, your schedule to fill the orders was not so efficient. You are now thinking of making the most efficient schedule, i.e. the schedule for all orders being filled in the shortest days, using the program you are going to write.

Input

The input consists of a series of test cases. Each case begins with a line containing a single integer N which represents the number of ordered tools.

Then N lines follow, each specifying an order by four elements separated by one or more space: *name*, day_1, *sup* and day_2. *name* is the name of the tool in the corresponding order. day_1 is the number of required days to make the tool without any support tool. *sup* is the name of the tool that can support carving the target tool. day_2 is the number of required days make the tool with the corresponding support tool.

The input terminates with the case where $N = 0$. You should not process this case.

You can assume that the input follows the constraints below:

- $N \leq 1000$;
- *name* consists of no longer than 32 non-space characters, and is unique in each case;
- *sup* is one of all *name*s given in the same case; and
- $0 < day_2 < day_1 < 20000$.

Output

For each test case, output a single line containing an integer which represents the minimum number of days required to fill all N orders.

Sample Input

```
3
gu 20  pa 10
ci 20  gu 10
pa 20  ci 10
2
tool 20 tool 10
product 1000 tool 10
8
fishhook  21  disk     3
mill      14  axe      7
boomerang 56  sundial 28
flint     35  fishhook 5
axe       35  flint   14
disk      42  disk     2
sundial   14  disk     7
hammer    28  mill     7
0
```

Output for the Sample Input

```
40
30
113
```

Problem E
Autocorrelation Function
Input: E.txt

Nathan O. Davis is taking a class of signal processing as a student in engineering. Today's topic of the class was autocorrelation. It is a mathematical tool for analysis of signals represented by functions or series of values. Autocorrelation gives correlation of a signal with itself. For a continuous real function $f(x)$, the autocorrelation function $R_f(r)$ is given by

$$\int_{-\infty}^{\infty} f(x)f(x+r)\,dx$$

where r is a real number.

The professor teaching in the class presented an assignment today. This assignment requires students to make plots of the autocorrelation functions for given functions. Each function is piecewise linear and continuous for all real numbers. The figure below depicts one of those functions.

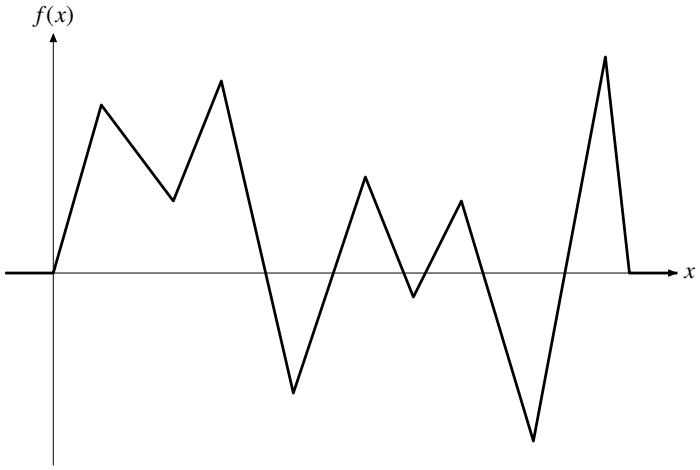

Figure 2: An Example of Given Functions

The professor suggested use of computer programs, but unfortunately Nathan hates programming. So he calls you for help, as he knows you are a great programmer. You are requested to write a program that computes the value of the autocorrelation function where a function $f(x)$ and a parameter r are given as the input. Since he is good at utilization of existing software, he could finish his assignment with your program.

Input

The input consists of a series of data sets.

The first line of each data set contains an integer n ($3 \leq n \leq 100$) and a real number r ($-100 \leq r \leq 100$), where n denotes the number of endpoints forming sub-intervals in the function $f(x)$, and r gives the parameter of the autocorrelation function $R_f(r)$. The i-th of the following n lines contains two integers x_i ($-100 \leq x_i \leq 100$) and y_i ($-50 \leq y_i \leq 50$), where (x_i, y_i) is the coordinates of the i-th endpoint. The endpoints are given in increasing order of their x-coordinates, and no couple of endpoints shares the same x-coordinate value. The y-coordinates of the first and last endpoints are always zero.

The end of input is indicated by $n = r = 0$. This is not part of data sets, and hence should not be processed.

Output

For each data set, your program should print the value of the autocorrelation function in a line. Each value may be printed with an arbitrary number of digits after the decimal point, but should not contain an error greater than 10^{-4}.

Sample Input

```
3 1
0 0
1 1
2 0
11 1.5
0 0
2 7
5 3
7 8
10 -5
13 4
15 -1
17 3
20 -7
23 9
24 0
0 0
```

Output for the Sample Input

```
0.166666666666667
165.213541666667
```

Problem F
It Prefokery Pio
Input: F.txt

You are a member of a secret society named *Japanese Abekobe Group*, which is called J. A. G. for short. Those who belong to this society often exchange encrypted messages. You received lots of encrypted messages this morning, and you tried to decrypt them as usual. However, because you received so many and long messages today, you had to give up decrypting them by yourself. So you decided to decrypt them with the help of a computer program that you are going to write.

The encryption scheme in J. A. G. utilizes palindromes. A palindrome is a word or phrase that reads the same in either direction. For example, "MADAM", "REVIVER" and "SUCCUS" are palindromes, while "ADAM", "REVENGE" and "SOCCER" are not.

Specifically to this scheme, words and phrases made of only one or two letters are not regarded as palindromes. For example, "A" and "MM" are not regarded as palindromes.

The encryption scheme is quite simple: each message is encrypted by inserting extra letters in such a way that the longest one among all subsequences forming palindromes gives the original message. In other words, you can decrypt the message by extracting the *longest palindrome subsequence*. Here, a subsequence means a new string obtained by picking up some letters from a string without altering the relative positions of the remaining letters. For example, the longest palindrome subsequence of a string "YMAOKDOAMIMHAADAMMA" is "MADAMIMADAM" as indicated below by underline.

YMAOKDOAMIMHAADAMMA

Now you are ready for writing a program.

Input

The input consists of a series of data sets. Each data set contains a line made of up to 2,000 capital letters which represents an encrypted string.

The end of the input is indicated by EOF (end-of-file marker).

Output

For each data set, write a decrypted message in a separate line. If there is more than one decrypted message possible (i.e. if there is more than one palindrome subsequence not shorter than any other ones), you may write any of the possible messages.

You can assume that the length of the longest palindrome subsequence of each data set is always longer than two letters.

Sample Input

```
YMAOKDOAMIMHAADAMMA
LELVEEL
```

Output for the Sample Input

```
MADAMIMADAM
LEVEL
```

Problem G
Traffic
Input: G.txt

You are a resident of *Kyoot* (oh, well, it's not a misspelling!) city. All streets there are neatly built on a grid; some streets run in a meridional (north-south) direction and others in a zonal (east-west) direction. The streets that run from north to south are called *avenues*, whereas those which run from east to west are called *drives*.

Every avenue and drive in the city is numbered to distinguish one from another. The westernmost avenue is called the *1st avenue*. The avenue which lies next to the *1st avenue* is the *2nd avenue*, and so forth. Similarly, drives are numbered from south to north. The figure below illustrates this situation.

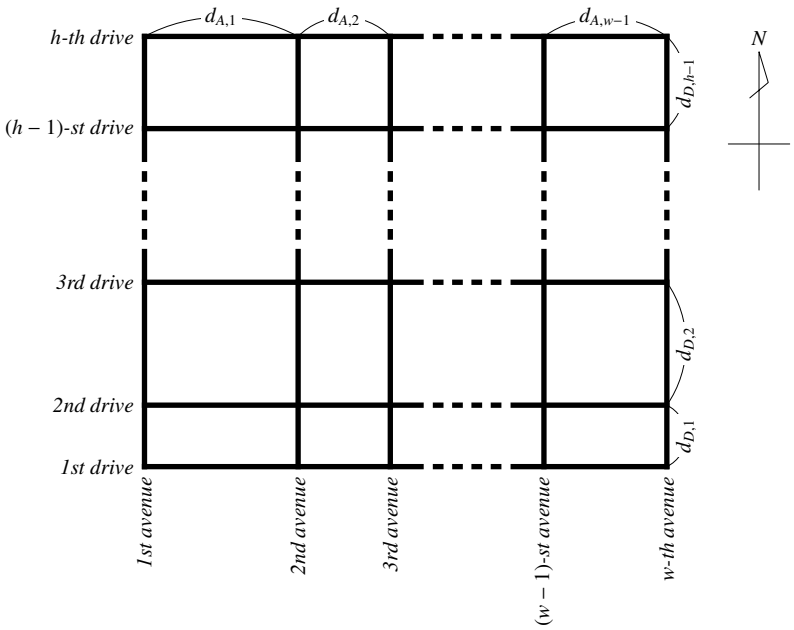

Figure 3: The Road Map of the Kyoot City

There is an intersection with traffic signals to regulate traffic on each crossing point of an avenue and a drive. Each traffic signal in this city has two lights. One of these lights is colored green, and means "you may go". The other is red, and means "you must stop here". If you reached an intersection during the red light (including the case where the light turns to red on your arrival), you must stop and wait there until the light turns to green again. However, you do not have to wait in the case where the light has just turned to green when you arrived there.

Traffic signals are controlled by a computer so that the lights for two different directions always show different colors. Thus if the light for an avenue is green, then the light for a drive must be red, and vice versa. In order to avoid car crashes, they will never be green together. Nor will they be red together, for

efficiency. So all the signals at one intersection turn simultaneously; no sooner does one signal turn to red than the other turns to green. Each signal has a prescribed time interval and permanently repeats the cycle.

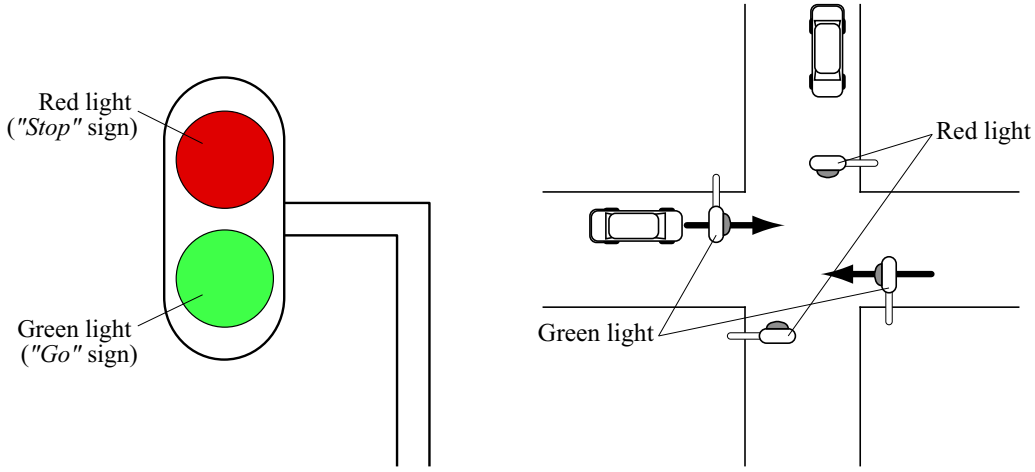

Figure 4: Signal and Intersection

By the way, you are planning to visit your friend by car tomorrow. You want to see her as early as possible, so you are going to drive through the shortest route. However, due to existence of the traffic signals, you cannot easily figure out which way to take (the city also has a very sophisticated camera network to prevent crime or violation: the police would surely arrest you if you didn't stop on the red light!). So you decided to write a program to calculate the shortest possible time to her house, given the town map and the configuration of all traffic signals.

Your car runs one unit distance in one unit time. Time needed to turn left or right, to begin moving, and to stop your car is negligible. You do not need to take other cars into consideration.

Input

The input consists of multiple test cases. Each test case is given in the format below:

$w\ h$
$d_{A,1}\ d_{A,2}\ \ldots\ d_{A,w-1}$
$d_{D,1}\ d_{D,2}\ \ldots\ d_{D,h-1}$
$ns_{1,1}\ ew_{1,1}\ s_{1,1}$
$\vdots$
$ns_{w,1}\ ew_{w,1}\ s_{w,1}$
$ns_{1,2}\ ew_{1,2}\ s_{1,2}$
$\vdots$
$ns_{w,h}\ ew_{w,h}\ s_{w,h}$
$x_s\ y_s$
$x_d\ y_d$

Two integers w and h ($2 \leq w, h \leq 100$) in the first line indicate the number of avenues and drives in the city, respectively. The next two lines contain $(w-1)$ and $(h-1)$ integers, each of which specifies the distance between two adjacent avenues and drives. It is guaranteed that $2 \leq d_{A,i}, d_{D,j} \leq 1,000$.

The next $(w \times h)$ lines are the configuration of traffic signals. Three parameters $ns_{i,j}$, $ew_{i,j}$ and $s_{i,j}$ describe the traffic signal (i, j) which stands at the crossing point of the i-th avenue and the j-th drive. $ns_{i,j}$ and $ew_{i,j}$ ($1 \leq ns_{i,j}, ew_{i,j} < 100$) are time intervals for which the light for a meridional direction and a zonal direction is green, respectively. $s_{i,j}$ is the initial state of the light: 0 indicates the light is green for a meridional direction, 1 for a zonal direction. All the traffic signal have just switched the state to $s_{i,j}$ at your departure time.

The last two lines of a case specify the position of your house (x_s, y_s) and your friend's house (x_d, y_d), respectively. These coordinates are relative to the traffic signal $(0, 0)$. x-axis is parallel to the drives, and y-axis is parallel to the avenues. x-coordinate increases as you go east, and y-coordinate increases as you go north. You may assume that these positions are on streets and will never coincide with any intersection.

All parameters are integers and separated by a blank character.

The end of input is identified by a line containing two zeros. This is not a part of the input and should not be processed.

Output

For each test case, output a line containing the shortest possible time from your house to your friend's house.

Sample Input

```
4 4
2 2 2
2 2 2
99 1 0
1 99 1
1 99 1
1 99 1
1 99 1
1 99 1
1 99 1
99 1 0
99 1 0
1 99 1
1 99 1
1 99 1
1 99 1
1 99 1
1 99 1
99 1 0
1 0
1 6
2 2
```

```
10
10
5 5 0
5 5 0
5 5 0
5 5 0
5 0
5 10
0 0
```

Output for the Sample Input

```
28
25
```

Problem H
Restriction Enzyme
Input: H.txt

Eye Ceapee-sea, is a scientist who works for ACM(Abnormal Computing Mechanism). She is developing a new intelligent computing system based on a genetic computing molecules. She believes that it is able to compute NP complete problems in polynomial time so that it will definitely impact the world in the near future.

However, there is a challange before her to complete the extraordinary system. Her system is based on complex molecules, which include genetic materials such as DNA. Therefore, they have a risk of mutation during proliferation. Her system now seems not working presumably because the DNA involved in the system has been crucially damaged. She has decided to make a genetic map called "Restriction enzyme map" to check which part of the DNA is damaged.

Restriction enzyme is an enzyme that cuts a DNA at the position where a particular short DNA sequence appears. For example, *SmaI* divides ACACAGGGCCCTCAGGTGC into two pieces, ACACAGGG and CCCTCAGGTGC. *SmaI* recognizes CCCGGG and cuts the sequence at the center. There are thousands types of restriction enzymes, each of which has its own target short sequence(s).

Restriction enzyme map is a map describing the names of restriction enzymes and the positions where the DNA sequence can be cut. An example is shown below. Note that the DNA below is circular (i.e. the left side and the right side of the sequence are connected).

```
Type                    SmaI        EcoRI       SmaI
DNA               >---------+-----------+-----------+----->
Relative position  0         5           11          17    20
```

Figure 5: Restriction Enzyme Map

If we can reconstruct the restriction enzyme map from some experimental information, she will compare the map of the original DNA with the reconstructed restriction enzyme map to find out the damaged region of the DNA.

Fortunately, she has a good (maybe expensive!) device which can measure the accurate lengths of the DNA fragments, so she has decided to use this device.

Her strategy is like this. First, she takes thousands of copies of the DNA from her computing system, then cuts them by restriction enzyme A, after which the length of the produced fragments are measured. Next, she takes another thousands of copies of the DNA, which are then cut by another restriction enzyme B, after which the lengths of the fragments are measured in the same manner. She also measures the lengths of the fragments cut by both enzyme A and B together. It is definitely true that those fragment lengths does not directly provide us with the restriction map itself, but we are often able to reconstruct the map using those lengths obtained by these three experiments.

Suppose A = enzyme *SmaI* and B = enzyme *EcoRI*.

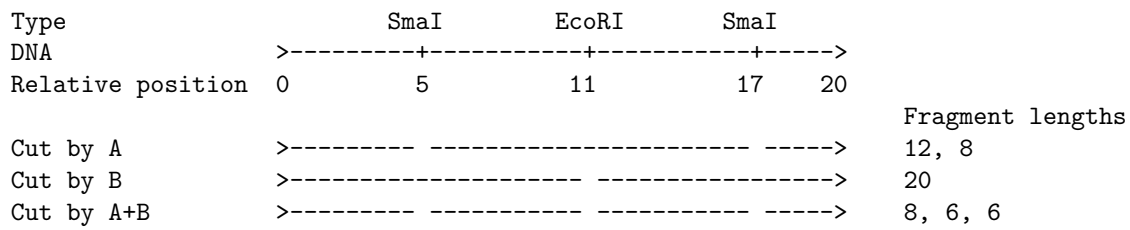

Figure 6: Cutting DNA by A, B, and $A + B$

Remind that the DNA is circular. You will get values 12 and 8 from the first experiment, where the DNA is cut by only enzyme A. When only enzyme B is used, the observed length of the fragment is 20, as this sequence has only one cutting site for *EcoRI*. You may assume at least one site exists for each enzyme. You may also assume the concentration of the enzymes is sufficiently high to cut every recognition site. When both enzyme A and B are used to cut the DNA, three fragments, each of which is 8, 6, 6, respectively in length, are produced. Although the measurement device tells us whether a fragment of specified length is contained in input fragments, it does not tell anything about how many types of fragments have a specified length. When more than one type of fragments is the same in length, only one value is reported, thus, in the last case, you will get only two values, 8 and 6, instead of three values, 8, 6 and 6.

Your task is to write a program to compute the restriction enzyme map from the information about the lengths of fragments cut by enzyme A, enzyme B, and enzyme $A + B$, respectively.

Input

The input consists of multiple cases. Each case is described by four lines. The first line is the length of the DNA. The lengths of the fragments cut by enzyme A are given in the second line. The first integer in the line represents the number of lengths to be given. The rest of the line consists of several integers each of which represents the length of a fragment. The lengths are separated by one space. Note that these values are not necessarily sorted. The third line and the last line consist of the lengths of the fragments cut by enzyme B and enzyme $A + B$, respectively. Although we do not know the exact recognition sequence of enzyme A nor B, you may assume that the cleavage at any sites do not interfere each other; in a general case, the cleavage by a certain restriction enzyme may break the recognition sequences of other enzymes, but you do not have to consider the case because Ms. Ceapee-sea guarantees that is not the case. She says that the recognition sequences of enzyme A and B differ, therefore, you may assume that no sites are recognized by both enzyme A and B at once.

The length of each DNA ranges from 2 to 20.

A line containing only '0' follows the last case.

Output

For each case, output the restriction enzyme map. Remind that the input DNA is circular. The first line of the case is n, the number of cutting sites. The following n lines describe the restriction enzyme map. Each line describes one cutting site, comprising of the relative position and the type of the restiction enzyme of the site. The relative position and the type of the restriction enzyme are separated by a single space, and no other characters must be included. The cutting sites should be given in ascending order.

If multiple maps suffice the constraint given by the input, choose one in which the number of cutting sites is not larger than the others. In case there still remains more than one map after applying this rule, you may choose any one of them.

The cases must be separated by a blank line.

Sample Input

```
6
2 2 4
1 3
2 1 2
10
2 2 4
1 2
2 2 1
0
```

Output for the Sample Input

```
4
0 A
1 B
2 A
4 B

8
0 A
1 B
2 A
3 B
5 B
6 A
7 B
9 B
```

Problem I
The Extreme Slalom
Input: I.txt

Automobile Company Moving has developed a new ride for switchbacks. The most impressive feature of this machine is its mobility that we can move and turn to any direction at the same speed.

As the promotion of the ride, the company started a new competition named *the extreme slalom*. A course of the extreme slalom consists of some gates numbered from 1. Each gate is represented by a line segment. A rider starts at the first gate and runs to the last gate by going through or touching the gates in the order of their numbers. Going through or touching gates other than the target one is allowed but not counted.

Your team is going to participate at the next competition. To win the competition, it is quite important to run on the shortest path. Your task is to write a program that computes the shortest length to support your colleagues.

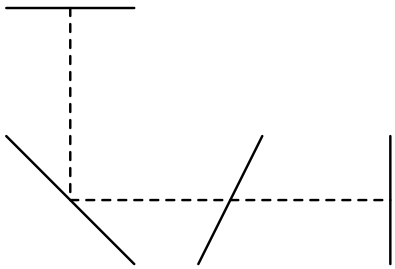

Figure 7: An Example Course for the Extreme Slalom

Input

The input consists of a number of courses.

The first line of a course is an integer indicating the number n ($2 \leq n \leq 12$) of gates. The following n lines specify the gates in the order of traversals. A line contains four integers x_1, y_1, x_2, and y_2 ($0 \leq x_1, y_1, x_2, y_2 \leq 100$) specifying the two endpoints of the gate. You may assume that no couple of gates touch or intersect each other.

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

Output

Print the shortest length out in one line for each course. You may print an arbitrary number of digits after the decimal points provided that difference from the exact answer is not greater than 0.0001.

Sample Input

```
4
0 4 2 4
0 2 2 0
3 0 4 2
6 2 6 0
0
```

Output for the Sample Input

```
8.0000
```