Problem Set for the 2nd Day of Summer Camp

Japanese Alumni Group

Contest Held on: 23 Sep 2006

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

Problem A
Moduic Squares
Input: A.txt

Have you ever heard of *Moduic Squares*? They are like 3×3 Magic Squares, but each of them has one extra cell called a *moduic cell*. Hence a Moduic Square has the following form.

A	B	C	
D	E	F	
G	H	I	J

Figure 1: A Moduic Square

Each of cells labeled from A to J contains one number from 1 to 10, where no two cells contain the same number. The sums of three numbers in the same rows, in the same columns, and in the diagonals in the 3×3 cells must be congruent modulo the number in the moduic cell. Here is an example Moduic Square:

3	1	6	
8	10	7	
9	4	2	5

Figure 2: An example Moduic Square

You can easily see that all the sums are congruent to 0 modulo 5.

Now, we can consider interesting puzzles in which we complete Moduic Squares with partially filled cells. For example, we can complete the following square by filling 4 into the empty cell on the left and 9 on the right. Alternatively, we can also complete the square by filling 9 on the left and 4 on the right. So this puzzle has two possible solutions.

3	1	6	
8	10	7	
		2	5

Figure 3: A Moduic Square as a puzzle

Your task is to write a program that determines the number of solutions for each given puzzle.

Input

The input contains multiple test cases. Each test case is specified on a single line made of 10 integers that represent cells A, B, C, D, E, F, G, H, I, and J as shown in the first figure of the problem statement. Positive integer means that the corresponding cell is already filled with the integer. Zero means that the corresponding cell is left empty.

The end of input is identified with a line containing ten of -1's. This is not part of test cases.

Output

For each test case, output a single integer indicating the number of solutions on a line. There may be cases with no solutions, in which you should output 0 as the number.

Sample Input

```
3 1 6 8 10 7 0 0 2 5
0 0 0 0 0 0 0 0 0 1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Output for the Sample Input

```
2
362880
```

Problem B
Restaurant
Input: B.txt

Steve runs a small restaurant in a city. Also, as he is the only cook in his restaurant, he cooks everything ordered by customers.

Basically he attends to orders by *first-come-first-served* principle. He takes some prescribed time to prepare each dish. As a customer can order more than one dish at a time, Steve starts his cooking with the dish that takes the longest time to prepare. In case that there are two or more dishes that take the same amount of time to prepare, he makes these dishes in the sequence as listed in the menu card of his restaurant. Note that he does not care in which order these dishes have been ordered. When he completes all dishes ordered by a customer, he immediately passes these dishes to the waitress and has them served to the customer. Time for serving is negligible.

On the other hand, during his cooking for someone's order, another customer may come and order dishes. For his efficiency, he decided to prepare together multiple dishes of the same if possible. When he starts cooking for new dishes, he looks over the orders accepted by that time (including the order accepted exactly at that time if any), and counts the number of the same dishes he is going to cook next. Time required for cooking is the same no matter how many dishes he prepare at once. Unfortunately, he has only limited capacity in the kitchen, and hence it is sometimes impossible that the requested number of dishes are prepared at once. In such cases, he just prepares as many dishes as possible.

Your task is to write a program that simulates the restaurant. Given a list of dishes in the menu card and orders from customers with their accepted times, your program should output a list of times when each customer is served.

Input

The input contains multiple data sets. Each data set is in the format below:

$N\ M$
$Name_1\ Limit_1\ Time_1$
...
$Name_N\ Limit_N\ Time_N$
$T_1\ K_1\ Dish_{1,1}\ \ldots\ Dish_{1,K_1}$
...
$T_M\ K_M\ Dish_{M,1}\ \ldots\ Dish_{M,K_M}$

Here, N ($1 \leq N \leq 20$) and M ($1 \leq M \leq 100$) specify the number of entries in the menu card and the number of orders that Steve will receive, respectively; each $Name_i$ is the name of a dish of the i-th entry in the menu card, which consists of up to 20 alphabetical letters; $Limit_i$ ($1 \leq Limit_i \leq 10$) is the number

of dishes he can prepare at the same time; $Time_i$ ($1 \leq Time_i \leq 1000$) is time required to prepare a dish (or dishes) of the i-th entry; T_j ($1 \leq T_j \leq 10000000$) is the time when the j-th order is accepted; K_j ($1 \leq K_j \leq 10$) is the number of dishes in the j-th order; and each $Dish_{j,k}$ represents a dish in the j-th order.

You may assume that every dish in the orders is listed in the menu card, but you should note that each order may contain multiple occurrences of the same dishes. The orders are given in ascending order by T_j, and no two orders are accepted at the same time.

The input is terminated with a line that contains two zeros. This is not part of data sets and hence should not be processed.

Output

Your program should produce the output of M-lines for each data set. The i-th line of the output should contain a single integer that indicates the time when the i-th order will be completed and served to the customer.

Print a blank line between two successive data sets.

Sample Input

```
5 4
Ramen 3 10
Chahan 5 5
Gyoza 5 10
Rice 1 1
Soup 1 1
5 2 Ramen Gyoza
10 6 Chahan Gyoza Soup Ramen Gyoza Rice
20 1 Chahan
25 1 Ramen
0 0
```

Output for the Sample Input

```
25
42
40
35
```

Problem C
Tetrahedra
Input: C.txt

Peter P. Pepper is facing a difficulty.

After fierce battles with the Principality of Croode, the Aaronbarc Kingdom, for which he serves, had the last laugh in the end. Peter had done great service in the war, and the king decided to give him a big reward. But alas, the mean king gave him a hard question to try his intelligence. Peter is given a number of sticks, and he is requested to form a tetrahedral vessel with these sticks. Then he will be, said the king, given as much *patas* (currency in this kingdom) as the volume of the vessel. In the situation, he has only to form a frame of a vessel.

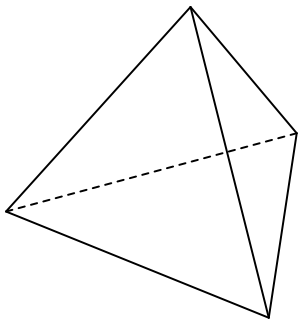

Figure 4: An example tetrahedron vessel

The king posed two rules to him in forming a vessel: (1) he need not use all of the given sticks, and (2) he must not glue two or more sticks together in order to make a longer stick.

Needless to say, he wants to obtain as much patas as possible. Thus he wants to know the maximum patas he can be given. So he called you, a friend of him, and asked to write a program to solve the problem.

Input

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

$$N\ a_1\ a_2\ \ldots\ a_N$$

where N indicates the number of sticks Peter is given, and a_i indicates the length (in centimeters) of each stick. You may assume $6 \leq N \leq 15$ and $1 \leq a_i \leq 100$.

The input terminates with the line containing a single zero.

Output

For each test case, print the maximum possible volume (in cubic centimeters) of a tetrahedral vessel which can be formed with the given sticks. You may print an arbitrary number of digits after the decimal point, provided that the printed value does not contain an error greater than 10^{-6}.

It is guaranteed that for each test case, at least one tetrahedral vessel can be formed.

Sample Input

```
7 1 2 2 2 2 2 2
0
```

Output for the Sample Input

```
0.942809
```

Problem D
International Party
Input: D.txt

Isaac H. Ives is attending an international student party (maybe for girl-hunting). Students there enjoy talking in groups with excellent foods and drinks. However, since students come to the party from all over the world, groups may not have a language spoken by all students of the group. In such groups, some student(s) need to work as interpreters, but intervals caused by interpretation make their talking less exciting.

Needless to say, students want exciting talks. To have talking exciting as much as possible, Isaac proposed the following rule: the number of languages used in talking should be as little as possible, and not exceed five. Many students agreed with his proposal, but it is not easy for them to find languages each student should speak. So he calls you for help.

Your task is to write a program that shows the minimum set of languages to make talking possible, given lists of languages each student speaks.

Input

The input consists of a series of data sets.

The first line of each data set contains two integers N ($1 \leq N \leq 30$) and M ($2 \leq M \leq 20$) separated by a blank, which represent the numbers of languages and students respectively. The following N lines contain language names, one name per line. The following M lines describe students in the group. The i-th line consists of an integer K_i that indicates the number of languages the i-th student speaks, and K_i language names separated by a single space. Each language name consists of up to twenty alphabetic letters.

A line that contains two zeros indicates the end of input and is not part of a data set.

Output

Print a line that contains the minimum number L of languages to be spoken, followed by L language names in any order. Each language name should be printed in a separate line. In case two or more sets of the same size is possible, you may print any one of them. If it is impossible for the group to enjoy talking with not more than five languages, you should print a single line that contains "Impossible" (without quotes).

Print an empty line between data sets.

Sample Input

```
3 4
English
```

```
French
Japanese
1 English
2 French English
2 Japanese English
1 Japanese
2 2
English
Japanese
1 English
1 Japanese
6 7
English
Dutch
German
French
Italian
Spanish
1 English
2 English Dutch
2 Dutch German
2 German French
2 French Italian
2 Italian Spanish
1 Spanish
0 0
```

Output for the Sample Input

```
2
English
Japanese

Impossible

Impossible
```

Problem E
Hide-and-seek
Input: E.txt

Hide-and-seek is a children's game. Players hide here and there, and one player called *it* tries to find all the other players.

Now you played *it* and found all the players, so it's turn to hide from *it*. Since you have got tired of running around for finding players, you don't want to play *it* again. So you are going to hide yourself at the place as far as possible from *it*. But where is that?

Your task is to find the place and calculate the maximum possible distance from *it* to the place to hide.

Input

The input contains a number of test cases.

The first line of each test case contains a positive integer N ($N \leq 1000$). The following N lines give the map where hide-and-seek is played. The map consists of N *corridors*. Each line contains four real numbers x_1, y_1, x_2, and y_2, where (x_1, y_1) and (x_2, y_2) indicate the two end points of the corridor. All corridors are straight, and their widths are negligible. After these N lines, there is a line containing two real numbers sx and sy, indicating the position of *it*. You can hide at an arbitrary place of any corridor, and *it* always walks along corridors. Numbers in the same line are separated by a single space.

It is guaranteed that *its* starting position (sx, sy) is located on some corridor and linked to all corridors directly or indirectly.

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

Output

For each test case, output a line containing the distance along the corridors from 'it"s starting position to the farthest position. The value may contain an error less than or equal to 0.001. You may print any number of digits below the decimal point.

Sample Input

```
2
0 0 3 3
0 4 3 1
1 1
0
```

Output for the Sample Input

```
4.243
```

Problem F
TV Watching
Input: F.txt

You are addicted to watching TV, and you watch so many TV programs every day. You have been in trouble recently: the airtimes of your favorite TV programs overlap.

Fortunately, you have both a TV and a video recorder at your home. You can therefore watch a program on air while another program (on a different channel) is recorded to a video at the same time. However, it is not easy to decide which programs should be watched on air or recorded to a video. As you are a talented computer programmer, you have decided to write a program to find the way of watching TV programs which gives you the greatest possible satisfaction.

Your program (for a computer) will be given TV listing of a day along with your *score* for each TV program. Each score represents how much you will be satisfied if you watch the corresponding TV program on air or with a video. Your program should compute the maximum possible sum of the scores of the TV programs that you can watch.

Input

The input consists of several scenarios.

The first line of each scenario contains an integer N ($1 \leq N \leq 1000$) that represents the number of programs. Each of the following N lines contains program information in the format below:

$$T\ T_b\ T_e\ R_1\ R_2$$

T is the title of the program and is composed by up to 32 alphabetical letters. T_b and T_e specify the start time and the end time of broadcasting, respectively. R_1 and R_2 indicate the score when you watch the program on air and when you have the program recorded, respectively.

All times given in the input have the form of "`hh:mm`" and range from 00:00 to 23:59. You may assume that no program is broadcast across the twelve midnight.

The end of the input is indicated by a line containing only a single zero. This is not part of scenarios.

Output

For each scenario, output the maximum possible score in a line.

Sample Input

```
4
OmoikkiriTV 12:00 13:00 5 1
WaratteIitomo 12:00 13:00 10 2
WatarusekennhaOnibakari 20:00 21:00 10 3
SuzumiyaharuhiNoYuuutsu 23:00 23:30 100 40
5
a 0:00 1:00 100 100
b 0:00 1:00 101 101
c 0:00 1:00 102 102
d 0:00 1:00 103 103
e 0:00 1:00 104 104
0
```

Output for the Sample Input

```
121
207
```

Problem G
Make Friendships
Input: G.txt

Isaac H. Ives attended an international student party and made a lot of girl friends (as many other persons expected). To strike up a good friendship with them, he decided to have dates with them. However, it is hard for him to schedule dates because he made so many friends. Thus he decided to find the best schedule using a computer program. The most important criterion in scheduling is how many different girl friends he will date. Of course, the more friends he will date, the better the schedule is. However, though he has the ability to write a program finding the best schedule, he doesn't have enough time to write it.

Your task is to write a program to find the best schedule instead of him.

Input

The input consists of a series of data sets. The first line of each data set contains a single positive integer N ($N \leq 1,000$) that represents the number of persons Isaac made friends with in the party. The next line gives Isaac's schedule, followed by N lines that give his new friends' schedules. Each schedule consists of a positive integer M that represents the number of available days followed by M positive integers each of which represents an available day.

The input is terminated by a line that contains a single zero. This is not part of data sets and should not be processed.

Output

For each data set, print a line that contains the maximum number of girl friends Isaac can have dates with.

Sample Input

```
3
3 1 3 5
2 1 4
4 1 2 3 6
1 3
0
```

Output for the Sample Input

```
2
```

Problem H
Slippy Floors
Input: H.txt

The princess of the Fancy Kingdom has been loved by many people for her lovely face. However the witch of the Snow World has been jealous of the princess being loved. For her jealousy, the witch has shut the princess into the Ice Tower built by the witch's extreme magical power.

As the Ice Tower is made of cubic ice blocks that stick hardly, the tower can be regarded to be composed of *levels* each of which is represented by a 2D grid map. The only way for the princess to escape from the tower is to reach downward stairs on every level. However, many magical traps set by the witch obstruct her moving ways. In addition, because of being made of ice, the floor is so slippy that movement of the princess is very restricted. To be precise, the princess can only press on one adjacent wall to move against the wall as the reaction. She is only allowed to move in the vertical or horizontal direction, not in the diagonal direction. Moreover, she is forced to keep moving without changing the direction until she is prevented from further moving by the wall, or she reaches a stairway or a magical trap. She must avoid the traps by any means because her being caught by any of these traps implies her immediate death by its magic. These restriction on her movement makes her escape from the tower impossible, so she would be frozen to death without any help.

You are a servant of the witch of the Snow World, but unlike the witch you love the princess as many other people do. So you decided to help her with her escape. However, if you would go help her directly in the Ice Tower, the witch would notice your act to cause your immediate death along with her. Considering this matter and your ability, all you can do for her is to generate *snowmans* at arbitrary places (grid cells) neither occupied by the traps nor specified as the cell where the princess is initially placed. These snowmans are equivalent to walls, thus the princess may utilize them for her escape. On the other hand, your magical power is limited, so you should generate the least number of snowmans needed for her escape.

You are required to write a program that solves placement of snowmans for given a map on each level, such that the number of generated snowmans are minimized.

Input

The first line of the input contains a single integer that represents the number of levels in the Ice Tower. Each level is given by a line containing two integers NY ($3 \leq NY \leq 30$) and NX ($3 \leq NX \leq 30$) that indicate the vertical and horizontal sizes of the grid map respectively, and following NY lines that specify the grid map. Each of the NY lines contain NX characters, where '`A`' represents the initial place of the princess, '`>`' the downward stairs, '`.`' a place outside the tower, '`_`' an ordinary floor, '`#`' a wall, and '`^`' a magical trap. Every map has exactly one '`A`' and one '`>`'.

You may assume that there is no way of the princess moving to places outside the tower or beyond the maps, and every case can be solved with not more than ten snowmans. The judge also guarantees that this problem can be solved without extraordinary optimizations.

Output

For each level, print on a line the least number of snowmans required to be generated for letting the princess get to the downward stairs.

Sample Input

```
3
10 30
......#############...........
....##_____________#..........
...#_________________####......
..#________####_________#.....
.#________#....##________#....
#_________#......#________#...
#__________###...#_____^^_#...
.#__________A_#...#____^^_#...
..#__________#...#_>__###....
...###########.....####.......
7 17
......#..........
.....#_##........
.#...#_A_#.......
#^####___#.......
#_______#........
#>_____#.........
########.........
6 5
#####
#_A_#
#___#
#_>_#
#___#
#####
```

Output for the Sample Input

```
2
1
0
```

Problem I
Roads in a City
Input: I.txt

Roads in a city play important roles in development of the city. Once a road is built, people start their living around the road. A broader road has a bigger capacity, that is, people can live on a wider area around the road.

Interstellar Conglomerate of Plantation and Colonization (ICPC) is now planning to develop roads on a new territory. The new territory is a square and is completely clear. ICPC has already made several plans, but there are some difficulties in judging which plan is the best.

Therefore, ICPC has collected several great programmers including you, and asked them to write programs that compute several metrics for each plan. Fortunately your task is a rather simple one. You are asked to compute the area where people can live from the given information about the locations and the capacities of the roads.

The following figure shows the first plan given as the sample input.

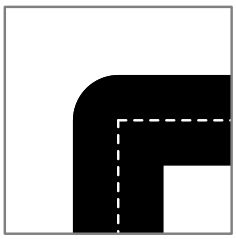

Figure 5: The first plan given as the sample input

Input

The input consists of a number of plans. The first line of each plan denotes the number n of roads ($n \leq 50$), and the following n lines provide five integers x_1, y_1, x_2, y_2, and r, separated by a space. (x_1, y_1) and (x_2, y_2) denote the two endpoints of a road ($-15 \leq x_1, y_1, x_2, y_2 \leq 15$), and the value r denotes the capacity that is represented by the maximum distance from the road where people can live ($r \leq 10$).

The end of the input is indicated by a line that contains only a single zero.

The territory is located at $-5 \leq x, y \leq 5$. You may assume that each road forms a straight line segment and that any lines do not degenerate.

Output

Print the area where people can live in one line for each plan. You may print an arbitrary number of digits after the decimal points, provided that difference from the exact answer is not greater than 0.01.

Sample Input

```
2
0 -12 0 0 2
0 0 12 0 2
0
```

Output for the Sample Input

```
39.14159
```