MIT Programming Contest 2002
Individual Round Problems

September 29, 2002

1 Cutting Boards

Farmer John recently bought a stack of boards to construct a fence around his cows. Unfortunately, the boards have different lengths. He knows the exact number of fence posts he needs for the fence and wants to cut the boards into fence posts in such a way so that the maximum difference between the height of any fence post and the average fence post height is minimized. In other words, let $b_1, b_2, \ldots b_n$ be the lengths of the boards that Farmer John bought and C be the number of fence posts Farmer John needs. Then find:

$$M = \min_{f_1,\ldots,f_C} \max_{1 \le i \le C} |f_i - \bar{f}|,$$

where $f_1, ..., f_C$ are the heights of the fence posts (for all i, f_i is a positive integer) and

$$\bar{f} = \frac{1}{C} \sum_{i=1}^{C} f_i.$$

Help Farmer John make his fence look pretty.

Input

The input data will consist of several test cases each representing a different stack of boards.

The first line of each set will contain two positive integer numbers N, C. N ($1 \le N \le 10000$) is the initial number of boards. C ($N \le C \le 1000000$) is the number of fence posts Farmer John needs for his fence. Each of the subsequent N lines contains one positive integer number b_i. $b_1, b_2, \ldots, b_N$ are the lengths of the boards ($1 \le b_i \le 10000$).

The input data is terminated by a line that contains a pair of zeroes, and should not be processed.

Output

For each test case, print out exactly one line containing a real number, denoting M, the minimum difference as described above. Print "Impossible" if it is impossible to cut the N boards to produce exactly C fence posts.

Formatting should be as in the following sample output. All numbers should be rounded to exactly two decimal places. The word "Done" should appear on the last output line.

Example

Sample Input

```
2 2
3
4
4 7
3
9
3
6
1 10
1
0 0
```

Sample Output

```
Stack 1
0.50

Stack 2
0.00

Stack 3
Impossible

Done
```

2 Largest Fence

Farmer John has built a large fence around his cows. The fence is made out of fence posts with tight barbed wire that wraps around all the fence posts. Farmer John is proud of the fact that his fence is reasonably resource-efficient, in that no fence post can be removed to increase the enclosed area of the fence.

Cows, being big creatures, like to have a lot of room to graze, so they were quite upset to discover one day that Farmer John had removed one of the fence posts because he was too lazy to go chop firewood in this rainy September. The next day, he took another fence post. The cows held a emergency meeting and decided that if each cow is no longer able to have at least one square yard of grazing space to herself, or if Farmer John keeps commiting these cowardly acts for a full month (30 days), whichever came first, then they would rebel.

Farmer John overheard the cows' conversation, and decided he will push his luck and stop taking the cows' fence posts when taking another fence post would mean rebellion. But he still wants to take as many fence posts as he can, so he must carefully plan which fence posts to remove each day. Since Farmer John hates programming even more than he hates chopping firewood, you must help him write this program.

Input

There will be several groups of input data, each representing a farm fence.

The first line of each set will contain two positive integer numbers N, C, where N ($3 \leq N \leq 100$) is the initial number of fence posts and C ($0 \leq C \leq 1000000$) is the number of cows enclosed by the fence. Each of the subsequent N lines contains two integer numbers x_i, y_i, which represent the integer coordinates (in yards) of the fence post locations, $-10000 \leq x_i, y_i \leq 10000$. These coordinates are listed in either clockwise or counter-clockwise order around the fence.

The input data is terminated by a line that contains zeroes for all input values, and should not be processed.

Output

The output should be labeled by the number of the fence. The following line should contain the maximum number of fence posts Farmer John can remove without causing a rebellion, along with the maximum possible area of the altered fence. If the cows don't have enough space in the initial fence, the line should read "Unhappy cows".

Formatting should be as in the following sample output. All numbers expressing areas should be rounded to exactly two decimal places. The word "Done" should appear on the last output line.

Example

Sample Input

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

Sample Output

```
Fence 1
Removed 0 fence posts, remaining area: 6.00

Fence 2
Unhappy cows

Fence 3
Removed 2 fence posts, remaining area: 2.00

Done
```

3 Cow Mail

Farmer John came up with a new way to keep his cows even without a fence. The idea was simple and genial — every day he gave the cows shoes of different height. Since then instead of walking in straight lines, they started walking in circles without ever going far away from the farm.

Farmer John was so happy with himself that he spend all the money he saved from the fence on buying PDAs for the cows. Cows, being very cowmunicative creatures, quickly started exchanging emails. The only problem was that their PDAs could connect to each other via the infrared port only when they were in a short distance from one another. Nevertheless, each cow was able to send an email to any other by routing it through other cows. Each message is tagged with routing information for the cows that it will need to go through. All email exchanges are done instantaneously when two cows are at the same location. All cows advance forward with constant speed of one foot per second. Whenever a cow needs to transfer email to another one, it advances to a point where their paths cross and waits there for the other. After a few days, Farmer John realized that quite often two cows wait indefinitely long for each other at different points where their circular paths cross. After that he made sure that two cows meet at no more than one point. (Of course, that didn't solve the rare cases where three or more cows wait for one another, but it appeared that they mooove on after several hours of grazing at one place.)

Your task is to write the mail routing software for the PDAs that finds the fastest route for message delivery between two cows. The decision should be based on the assumption that the expected time for a cow to reach a point on its path is equal to the time it takes it to go half around its circle.

Input

There will be several groups of input data, each representing a day in the farm.

The first line of each set will contain three positive integer numbers n, i, j, representing the number of cows, $1 \leq n \leq 100$, and the source i and destination j of the message, $1 \leq i, j \leq n$. Each of the subsequent n lines contains three positive real numbers x_i, y_i, r_i which represent the circle walked by cow with identifier i. x, y are the coordinates of the center, and r is the radius, $1 \leq x, y, r \leq 1000$. All cows with odd identifiers i walk clockwise, while the cows with even identifiers walk counterclockwise.

The input data is terminated by a line that contains zeroes for all input values, and should not be processed.

Output

The output for each data set should be labeled by the number of the day, and followed by a line specifying the expected delivery time in seconds for a message if target is reachable, otherwise print "Message delivery not possible".

Formatting should be as in the following sample output. All numbers should be rounded to exactly two decimal places. The word "Done" should appear on the last output line.

Example

Sample Input

```
4 1 3
10 10 5
10 20 5
20 20 5
20 10 5
2 1 2
1.0 1.0 5.0
1.0 9.0 3.0
2 2 1
1.1 2.2 5.5
9.9 9.9 3.14
0 0 0
```

Sample Output

```
Day 1
Expected delivery time 54.98

Day 2
Expected delivery time 25.13

Day 3
Message delivery not possible

Done
```

4 Greenspeak

As part of its Rush campaign the Alpha Chi Mu fraternity is preparing a "Greenspeak" slide show. "Greenspeak" in MIT hackers lingo means making a pattern out of lights in the Green Building (Building 54) – the 23–floor building is the tallest at MIT and Cambridge. Each of the slides is constituted of pixels representing the lights on the rectangular grid of the Green building rooms facing the Charles river. ACM put together their masterpiece of enticing verbiage and graphics, with the plan to quickly set on and off the room lights to show their low resolution but prepared with high devotion slides.

Unfortunately, rumors about their grand plan leak out. The administration comes with the latest adendum to Tute rules that forbids switching a room lights on and off more than ϵ times per hour.

Since it is too late to come up with another great idea, they decide to skip some slides to keep the show running without violating the new rule. The whole event is taking less than an hour, therefore the condition is simply that no room light is changing state more than ϵ times for the whole set. Assume that all lights are off at the time the slide show starts, and should be turned off at the end. Of course, they also want to minimize the number of slides skipped.

It seems like they will need your help to quickly determine which slides to skip.

Input

There will be several groups of input data, each representing a slide show.

The first line of each set will contain four positive integer numbers m, n, s and ϵ. m and n are the dimensions of a subset of the rectangular grid of Building 54, $1 \leq m \leq 23, 1 \leq n \leq 9$. s is the number of original slides , ϵ is as defined above the maximum number of state changes for a room light.

The next s data blocks represent the slides in row major mode. Each slide is depicted in m lines containing n binary digits separated by spaces. The value 1 represents a lit room, while 0 — unlit.

The input data is terminated by a line that contains zeroes for all input values, and should not be processed.

Output

The output should be labeled by the number of the slide show, and followed by a line containing the list of slides that need to be skipped, or "Slide show conforms to the new rule" if none have to be skipped. The number of slides in the list has to be minimal, and if more than one minimal length lists are possible the lexicographically minimal one should be printed with slide numbers delimited by spaces.

Formatting should be as in the following sample output. The word "Done" should appear on the last output line.

Example

Sample Input

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

Sample Output

```
Slide show 1
Slide show conforms to the new rule

Slide show 2
Skip slides: 1 2

Done
```

5 Contest

A restructuring of the Annual Computing Marathon (ACM) has taken place due too the persistent complaints of contestants. They complained about the evilness of computational geometry problems; those have now been banned. They complained about the pages of irrelevant background information in the problem descriptions; descriptions are now limited to five sentences. They complained that it was humanly impossible to solve all nine problems; now, the number of problems has been reduced to three. (This would allow all teams to solve all the problems and give the teams that finished early a chance to heckle the slower teams, making the event more feisty and more interesting to spectators. As a result, the word "Marathon" in ACM has been replaced with "Melee.")

But there were still complaints. There would be teams that solved one problem really quickly but did not solve the other two problems. They would be upset that they were not recognized for their excellence on that one problem. To remedy this situation, contest organizers decided that they would award a medal to every "respectable" team, where a team is "respectable" if no other team thrashed it. A team T_1 thrashes a team T_2 if, for each problem, T_1 solves the problem at least as fast as T_2, and there's some problem that T_1 solved faster than T_2.

The contest organizers want you to write an efficient program to count the number of medals they need to order, given the results of the contest.

Input

There will be several groups of input data, each describing a single contest.

The first line of each set will contain a positive integer number N ($1 \leq N \leq 500000$) representing the number of teams. Each of the subsequent N lines contains three positive integer numbers a_i, b_i, c_i ($1 \leq a_i, b_i, c_i \leq 1000$) denoting the times in seconds that it took team i to solve the first, second, and third problems, respectively.

The input data is terminated by a line that contains a single zero, and should not be processed.

Output

The output for each data set should be labeled by a contest number. For each test case, print one line indicating the number of medals needed.

Formatting should be as in the following sample output. The word "Done" should appear on the last output line.

Example

Sample Input

```
1
3 5 6
2
5 5 5
4 4 5
6
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
7
1 1 1
1 1 2
1 2 2
2 2 2
3 2 2
3 3 2
3 3 3
0
```

Sample Output

```
Contest 1
1 medals needed

Contest 2
1 medals needed

Contest 3
6 medals needed

Contest 4
1 medals needed

Done
```

6 Dessert

At the end of another free food event, our regular attendee Ben Bitdiddle is standing in front the perpetual dilemma: *apple pie*, *lemon cake*, or maybe both? After a moment of hesitation, Ben is almost ready to take both, then he sees *chocolate brownies* coming to the table and he starts pondering again what to take. Ben doesn't want to make a bad impression, so he decides he will take only one plate. Furthermore, as a free food connoisseur, he doesn't want to mix flavours by putting desserts on top of each other. All pieces of dessert have the shape of right-angled triangles as result of halving rectangular pieces. Each piece is served in a circular shape plate that tightly fits one piece.

Ben quickly takes out his notebook and starts coding furiously to decide whether he can fit pieces of the three desserts in one plate. You have to help Ben to solve the problem before all desserts are gone.

Input

There will be several groups of input data, each representing a dessert menu.

Each line of data will contain three dessert specifications given as a pair of positive real numbers. Each pair represents the lengths of the two sides of the rectangle which is halved to give the triangular pieces.

The input data is terminated by a line that contains zeroes for all input values, and should not be processed.

Output

The output should be labeled by the number of the menu, and followed by a line indicating whether or not three different desserts can fit in one plate.

Formatting should be as in the following sample output. The word "Done" should appear on the last output line.

Example

Sample Input

```
3.0 4.0 3.0 4.0 1.0 10.0
1.0 1.0 1.0 1.0 1.0 1.0
0.0 0.0 0.0 0.0 0.0 0.0
```

Sample Output

```
Menu 1
Three different pieces can fit in one plate

Menu 2
You can't sample all with one plate

Done
```

This page intentionally left blank.