Problem Set for the 1st Day of Summer Camp 2007

Japanese Alumni Group

Contest Held on 22 Sep 2007

Status of Problems

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

Some portion of the problem statement was modified for clarifications.

Terms of Use

Problem A
Dance Dance Revolution

Input: A.txt

Summer Camp (Day 1), Tokyo
22 Sep 2007

Dance Dance Revolution is one of the most popular arcade games in Japan. The rule of this game is very simple. A series of four arrow symbols, *up*, *down*, *left* and *right*, flows downwards on the screen in time to music. The machine has four panels under your foot, each of which corresponds to one of the four arrows, and you have to make steps on these panels according to the arrows displayed on the screen. The more accurate in timing, the higher score you will mark.

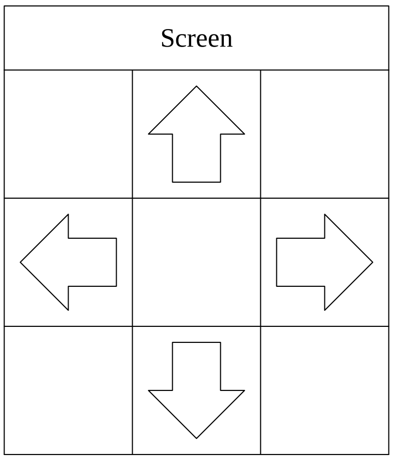

Figure 1: Layout of Arrow Panels and Screen

Each series of arrows is commonly called a *score*. Difficult scores usually have hundreds of arrows, and newbies will often be scared when seeing those arrows fill up the monitor screen. Conversely, scores with fewer arrows are considered to be easy ones.

One day, a friend of yours, who is an enthusiast of this game, asked you a favor. What he wanted is an automated method to see if given scores are *natural*. A score is considered as natural when there is a sequence of steps that meets all the following conditions:

- left-foot steps and right-foot steps appear in turn;
- the same panel is not stepped on any two consecutive arrows;
- a player can keep his or her upper body facing forward during a play; and
- his or her legs never cross each other.

Your task is to write a program that determines whether scores are natural ones.

► Input

The first line of the input contains a positive integer. This indicates the number of data sets.

Each data set consists of a string written in a line. Each string is made of four letters, "`U`" for up, "`D`" for down, "`L`" for left and "`R`" for right, and specifies arrows in a score. No string contains more than 100,000 characters.

► Output

For each data set, output in a line “`Yes`” if the score is natural, or “`No`” otherwise.

► Sample Input

```
3
UU
RDUL
ULDURDULDURDULDURDULDURD
```

► Output for the Sample Input

```
No
Yes
Yes
```

Problem B
Compress Files

Input: B.txt

Summer Camp (Day 1), Tokyo
22 Sep 2007

Your computer is a little old-fashioned. Its CPU is slow, its memory is not enough, and its hard drive is near to running out of space. It is natural for you to hunger for a new computer, but sadly you are not so rich. You have to live with the aged computer for a while.

At present, you have a trouble that requires a temporary measure immediately. You downloaded a new software from the Internet, but failed to install due to the lack of space. For this reason, you have decided to compress all the existing files into archives and remove all the original uncompressed files, so more space becomes available in your hard drive.

It is a little complicated task to do this within the limited space of your hard drive. You are planning to make free space by repeating the following three-step procedure:

1. Choose a set of files that have not been compressed yet.
2. Compress the chosen files into one new archive and save it in your hard drive. Note that this step needs space enough to store both of the original files and the archive in your hard drive.
3. Remove all the files that have been compressed.

For simplicity, you don't take into account extraction of any archives, but you would like to reduce the number of archives as much as possible under this condition. Your task is to write a program to find the minimum number of archives for each given set of uncompressed files.

► Input

The input consists of multiple data sets.

Each data set starts with a line containing two integers n ($1 \leq n \leq 14$) and m ($1 \leq m \leq 1000$), where n indicates the number of files and m indicates the available space of your hard drive before the task of compression. This line is followed by n lines, each of which contains two integers b_i and a_i. b_i indicates the size of the i-th file without compression, and a_i indicates the size when compressed. The size of each archive is the sum of the compressed sizes of its contents. It is guaranteed that $b_i \geq a_i$ holds for any $1 \leq i \leq n$.

A line containing two zeros indicates the end of input.

► Output

For each test case, print the minimum number of compressed files in a line. If you cannot compress all the files in any way, print "`Impossible`" instead.

► Sample Input

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

► Output for the Sample Input

```
2
Impossible
```

Problem C
Save the Energy

Input: C.txt

Summer Camp (Day 1), Tokyo
22 Sep 2007

You were caught in a magical trap and transferred to a strange field due to its cause. This field is three-dimensional and has many straight paths of infinite length. With your special ability, you found where you can exit the field, but moving there is not so easy. You can move along the paths easily without your energy, but you need to spend your energy when moving outside the paths. One unit of energy is required per distance unit. As you want to save your energy, you have decided to find the best route to the exit with the assistance of your computer.

Your task is to write a program that computes the minimum amount of energy required to move between the given source and destination. The width of each path is small enough to be negligible, and so is your size.

► Input

The input consists of multiple data sets. Each data set has the following format:

N
$x_s\ y_s\ z_s\ x_t\ y_t\ z_t$
$x_{1,1}\ y_{1,1}\ z_{1,1}\ x_{1,2}\ y_{1,2}\ z_{1,2}$
$\vdots$
$x_{N,1}\ y_{N,1}\ z_{N,1}\ x_{N,2}\ y_{N,2}\ z_{N,2}$

N is an integer that indicates the number of the straight paths ($2 \leq N \leq 100$). (x_s, y_s, z_s) and (x_t, y_t, z_t) denote the coordinates of the source and the destination respectively. $(x_{i,1}, y_{i,1}, z_{i,1})$ and $(x_{i,2}, y_{i,2}, z_{i,2})$ denote the coordinates of the two points that the i-th straight path passes. All coordinates do not exceed 30,000 in their absolute values.

The distance units between two points (x_u, y_u, z_u) and (x_v, y_v, z_v) is given by the Euclidean distance as follows:

$$\sqrt{(x_v - x_u)^2 + (y_v - y_u)^2 + (z_v - z_u)^2}.$$

It is guaranteed that the source and the destination both lie on paths. Also, each data set contains no straight paths that are almost but not actually parallel, although may contain some straight paths that are strictly parallel.

The end of input is indicated by a line with a single zero. This is not part of any data set.

► Output

For each data set, print the required energy on a line. Each value may be printed with an arbitrary number of decimal digits, but should not contain the error greater than 0.001.

► Sample Input

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

► Output for the Sample Input

```
1.414
2.000
3.000
```

Problem D
Goofy Converter

Input: D.txt

Summer Camp (Day 1), Tokyo
22 Sep 2007

Nathan O. Davis is a student at the department of integrated systems. He is now taking a class in integrated curcuits. He is an idiot. One day, he got an assignment as follows: design a logic circuit that takes a sequence of positive integers as input, and that outputs a sequence of 1-bit integers from which the original input sequence can be restored uniquely.

Nathan has no idea. So he searched for hints on the Internet, and found several pages that describe the 1-bit DAC. This is a type of digital-analog converter which takes a sequence of positive integers as input, and outputs a sequence of 1-bit integers.

Seeing how 1-bit DAC works on these pages, Nathan came up with a new idea for the desired converter. His converter takes a sequence L of positive integers, and a positive integer M aside from the sequence, and outputs a sequence K of 1-bit integers such that:

$$L_j = \sum_{i=j}^{j+M-1} K_i.$$

He is not so smart, however. It is clear that his converter does not work for some sequences. Your task is to write a program in order to show the new converter cannot satisfy the requirements of his assignment, even though it would make Nathan in despair.

► Input

The input consists of a series of data sets. Each data set is given in the following format:

N M
L_0 L_1 $\dots$ L_{N-1}

N is the length of the sequence L. M and L are the input to Nathan's converter as described above. You may assume the followings: $1 \le N \le 1000$, $1 \le M \le 12$, and $0 \le L_j \le M$ for $j = 0, \dots, N-1$.

The input is terminated by $N = M = 0$.

► Output

For each data set, output a binary sequence K of the length $(N + M - 1)$ if there exists a sequence which holds the equation mentioned above, or "`Goofy`" (without quotes) otherwise. If more than one sequence is possible, output any one of them.

► Sample Input

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

► Output for the Sample Input

```
1111001
Goofy
```

Problem E
Black Force

Input: E.txt

Summer Camp (Day 1), Tokyo
22 Sep 2007

A dam construction project was designed around an area called Black Force. The area is surrounded by mountains and its rugged terrain is said to be very suitable for constructing a dam.

However, the project is now almost pushed into cancellation by a strong protest campaign started by the local residents. Your task is to plan out a compromise proposal. In other words, you must find a way to build a dam with sufficient capacity, without destroying the inhabited area of the residents.

The map of Black Force is given as $H \times W$ cells ($0 < H, W \leq 20$). Each cell $h_{i,j}$ is a positive integer representing the height of the place. The dam can be constructed at a connected region surrounded by higher cells, as long as the region contains neither the outermost cells nor the inhabited area of the residents. Here, a region is said to be connected if one can go between any pair of cells in the region by following a sequence of left-, right-, top-, or bottom-adjacent cells without leaving the region. The constructed dam can store water up to the height of the lowest surrounding cell. The capacity of the dam is the maximum volume of water it can store. Water of the depth of 1 poured to a single cell has the volume of 1.

The important thing is that, in the case it is difficult to build a sufficient large dam, it is allowed to choose (at most) one cell and do groundwork to increase the height of the cell by 1 unit. Unfortunately, considering the protest campaign, groundwork of larger scale is impossible. Needless to say, you cannot do the groundwork at the inhabited cell.

Given the map, the required capacity, and the list of cells inhabited, please determine whether it is possible to construct a dam.

► Input

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

$H\ W\ C\ R$
$h_{1,1}\ h_{1,2} \ldots h_{1,W}$
$\ldots$
$h_{H,1}\ h_{H,2} \ldots h_{H,W}$
$y_1\ x_1$
$\ldots$
$y_R\ x_R$

H and W is the size of the map. C is the required capacity. R ($0 < R < H \times W$) is the number of cells inhabited. The following H lines represent the map, where each line contains W numbers separated by space. Then, the R lines containing the coordinates of inhabited cells follow. The line "$y\ x$" means that the cell $h_{y,x}$ is inhabited.

The end of input is indicated by a line "`0 0 0 0`". This line should not be processed.

► Output

For each data set, print "`Yes`" if it is possible to construct a dam with capacity equal to or more than C. Otherwise, print "`No`".

► Sample Input

```
4 4 1 1
2 2 2 2
2 1 1 2
2 1 1 2
2 1 2 2
1 1
4 4 1 1
2 2 2 2
2 1 1 2
2 1 1 2
2 1 2 2
2 2
4 4 1 1
2 2 2 2
2 1 1 2
2 1 1 2
2 1 1 2
1 1
3 6 6 1
1 6 7 1 7 1
5 1 2 8 1 6
1 4 3 1 5 1
1 4
5 6 21 1
1 3 3 3 3 1
3 1 1 1 1 3
3 1 1 3 2 2
3 1 1 1 1 3
1 3 3 3 3 1
3 4
0 0 0 0
```

► Output for the Sample Input

```
Yes
No
No
No
Yes
```