UNB Spring 2010 Programming Contest

Problem Descriptions

March 13, 2010

Sponsored by

A Next!

by Nathan Scott

Jordan has a very important appointment this morning, and has dutifully arrived very early for it. Unfortunately it seems as though everyone who has an appointment this morning also arrived early. Jordan has been forced to take a number and await his turn with everyone else in the waiting room. What is worse is that Jordan can smell the sweet aroma of hash browns and coffee wafting from the nearby cafeteria! He really wants to enjoy an early morning coffee, but he also knows that if he is not in the waiting room when his number is called he will have to begin waiting all over again. He is not allowed to leave the cafeteria with his food or drink, the rules strictly prohibit such substances in the hallways or waiting room. Fortunately, he has some information to work with. He has seen several numbers called already, and so can estimate how long an appointment takes. He has had many a cup of coffee in his lifetime and so can estimate how long his cafeteria trip will take. He also knows his own number and the most recent number called. Write a program to advise Jordan as to whether or not he has enough time to go get coffee or not.

Input

The first line of the input contains a number $n > 0$ that represents the number of test cases that follow. n lines follow, each reperesenting a test case. A test case consists of 4 integers a, c, i, j. a is the length of an appointment. c is the time it will take Jordan to have his coffee (including travel to and from the cafeteria). i is the most recent number called for an appointment. j is Jordans number. All times are in minutes, all numbers are positive, and $j \leq i$.

Output

Each test case should result in a single line of output. If there is enough time for Jordan to have his coffee before his number is called, output "Go for it!" Otherwise, output "Not enough time." You may assume the decision is made the very instant the number i is called, and that all numbers between i and j will be called in order. Jordan will not miss his appointment if he gets back to the waiting room at the exact instant his number is called.

Sample Input

```
4
5 10 103 106
5 16 15 18
20 19 99 100
15 5 1234 1234
```

Sample Output

```
Go for it!
Not enough time.
Go for it!
Not enough time.
```

B Run Forrest Run!

by Ken Kent

Running is a fascinating sport when it comes to statistics. Runners are religious about keeping track of time and always trying to go farther in less time. Needless to say when a group of runners get together all of the talk is about how fast they are and how much time it would take them to travel some distance. Thankfully, a runner does not need to be after running all distances to be able to see roughly how well they can do. The following formula has been developed to predict a runner's time for running a marathon, given a time in another distance.

$$\text{time} = t_1 * (42.2/d_1)^{1.06}$$

To be precise, if one has several times for a runner then you can average the predicted times for a more accurate prediction.

Input

Input to your program will be provided through stdin. The first line will contain the number of running performances (at least 1). Following this on each line is the runners numeric id, the distance run, and the runners time for that distance in min:sec. For example the following sample file shows two runners. Runner #1 ran 5 kms in 22 minutes and 35 seconds. Runner #2 ran 10 kms in 46 minutes and 12 seconds. Runner #1 ran 21.1 kms (1/2 marathon) in 91 minutes and 7 seconds. Runner #3 ran 21.1 kms (1/2 marathon) in 137 minutes and 12 seconds.

```
4
1 5 22:35
2 10 46:12
1 21.1 91:07
3 21.1 137:12
```

Output

The output of the program will provide the predicted times for each runner in the finish order if they were to race. You can assume that no marathon will take place that has more than 10,000 competitors to ensure safe running conditions.

```
Race Results
1. 1 203:17
2. 2 212:33
3. 3 286:03
```

Note that seconds should always be printed with two digits.

C Are the towns connected by roads?

selected by Joe Horton

The Emergency Measures Organization for an unnamed state wishes to know quickly when a hurricane strikes which counties in the state still have roads connecting all the towns of the county together. You must write a program to determine this.

Input

On the first line there is a number indicating the number of counties. Following that, there is data for each of the counties. For each county, the first line contains the name of the county followed by the number specifying the number of towns in the county. Each subsequent line contains the names of two towns that are connected by a road. A blank line indicates that all other roads in the county have been washed out by the hurricane. You may assume that the names consist of a single word, that is, that the names do not contain any white space.

Output

for each county, in the order that the counties were input, write out one line stating either:

County (insert name) is connected.

or

County (insert name) is disconnected.

Sample input

```
2
One  3
First Second
Second First

Two 2
Third Fourth
```

Note the sample input ends with a blank line.

Sample Output

```
County One is disconnected.
County Two is connected.
```

D Triangles

by Patricia Evans

You are examining a set of images that contain triangles, and you are trying to match triangles that could be the same. The triangles in the images may have been scaled, flipped, and rotated. You need to partition the set of triangles into sets of similar triangles, ones that are the same triangle except for potential scaling, flipping, and rotating. Measurements have been taken of all triangles, so for each of them you have a list of the lengths of their sides, given clockwise.

Input

The first line of the input gives the number of test cases. Each case consists of an unknown number of triangles, each specified by their 3 side lengths in integers, with each triangle on a separate line and the case ending with three 0-length sides.

Output

For each case, list the sets of similar triangles, with one set per line and the triangles in a set separated by commas. Each set should be in nondecreasing order of triangle size, and the sets should be given in nonincreasing order of set size. All ties are broken by lexicographic order. There should be a blank line between each pair of successive cases.

Sample Input

```
3
2 1 3
1 1 1
5 3 1
10 2 6
4 2 6
4 12 8
1 3 5
0 0 0
15 5 5
4 10 6
5 2 3
1 3 1
2 2 6
0 0 0
6 6 1
6 2 6
0 0 0
```

Sample Output

```
1 3 5, 5 3 1, 10 2 6
2 1 3, 4 2 6, 4 12 8
1 1 1

1 3 1, 2 2 6, 15 5 5
5 2 3, 4 10 6

6 2 6
6 6 1
```

E Factorial Divisibility

selected by Joe Horton

The number ($n!$) is defined to be the product of the n smallest positive integers. Thus $4! = 1 \times 2 \times 3 \times 4$. We want to know how many times you can divide 2 into ($n!$). Since 2 divides 12 once, and 2 divides 4 twice, but does not divide 3 at all, the number of times 2 divides ($4!$) is exactly three. Write a program to determine the number of times $n!$ is divisible by 2.

Input

A sequence of positive integers all on one line, separated by spaces, and followed by 0.

Output

For each input integer n, excluding the last zero, print out a line: "n factorial is divisible by exactly k 2's."

Sample input

```
5 7 2 0
```

Output should be:

```
5 factorial is divisible by exactly 3 twos.
7 factorial is divisible by exactly 4 twos.
2 factorial is divisible by exactly 1 twos.
```

(Ignor the bad grammar in the special case when the number of twos is 1.)

F Incremental Numbers

by David Bremner

An *incremental number* is one where the first occurances of digits occur in order. The first occurance of 1 occurs before the first occurance of 2, 2 before 3, and so on.

1122321	is incremental; after the first occurance, the digits can occur in any order.
2123	is not incremental.

Input

One or more lines of the form

$k \quad n$

where k and n are positive integers, separated by one or more spaces. You may assume $1 \leq k \leq 9$ and $1 \leq n \leq 13$.

Output

For each input of k n, print all n digit incremental numbers using digits 1..k, in increasing order, each on a line by itself. Terminate the input for each case with 0, on a line by itself.

Sample Input

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

Sample Output

```
11
0
1112
1121
1122
1211
1212
1221
1222
0
1123
1213
1223
1231
1232
1233
0
```

G Clockworks

by Patricia Evans

A string is being threaded inside a grid of rotatable spools, from the upper left to the bottom right. At each spool, the thread needs to exit the spool either to the right or down. The string must be threaded around the spool in the direction that the spool rotates, and each spool rotates either clockwise or counterclockwise. Threading the string one quarter of the way around a spool requires one unit of length. A string, then, that enters the spool from the left and exits to the right uses 2 units for this spool; a string that enters a clockwise spool from the left and exits down uses 3 units, while if it enters a counterclockwise spool from the left and exits down it will only use 1 unit. Similarly, a string that enters from the top and exits down uses 2 units, a string that enters a clockwise spool from the top and exits right uses 1 unit, and a string that enters a counterclockwise spool from the top and exits right uses 3 units.

Your problem is to determine how long the string needs to be to thread from the upper left spool (entering from above) to the lower right spool (exiting down), by the minimum length route.

Input

The first line of input gives the number of test cases. For each case, the first line of each case gives n and m, the dimensions of the grid, both of which are no greater than 100. The remaining n lines consist of m symbols each, where each symbol is either N (indicating clockwise, turning to the right and down) or R (indicating counterclockwise, turning to the left and down).

Output

Each case should produce a separate line of output, giving the minimum length of string needed to thread through the spools.

Sample Input

```
2
3 2
NR
RN
NN
4 4
NRRN
RNNR
NNNN
RRRR
```

Sample Output

```
6
10
```