FIU | FLORIDA INTERNATIONAL UNIVERSITY

Ultimate SOFTWARE People first.

2013 High School Programming Competition

April 6, 2013

Problem Set

A: Box of Bricks

B: Complimentary Binary Strings

C: Counting in Base Seven

D: Dog and Gopher

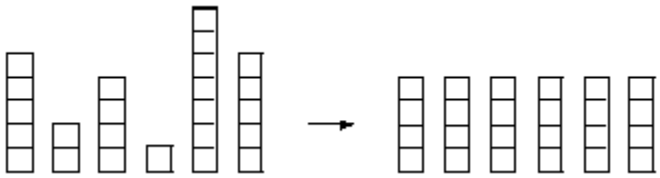
E: Freckles

F: The Tourist Guide

G: Word Counting

A: Box of Bricks

Little Bob likes playing with his box of bricks. He puts the bricks one upon another and builds stacks of different height. "Look, I've built a wall!", he tells his older sister Alice. "Nah, you should make all stacks the same height. Then you would have a real wall.", she retorts. After a little con- sideration, Bob sees that she is right. So he sets out to rearrange the bricks, one by one, such that all stacks are the same height afterwards. But since Bob is lazy he wants to do this with the minimum number of bricks moved. Can you help?



Input

The input consists of several data sets. Each set begins with a line containing the number *n* of stacks Bob has built. The next line contains *n* numbers, the heights h_i of the *n* stacks. You may assume 1 <= n <= 50 and 1 <= h_i <= 100. The total number of bricks will be divisible by the number of stacks. Thus, it is always possible to rearrange the bricks such that all stacks have the same height.

The input is terminated by a set starting with *n* = 0. This set should not be processed.

Output

For each set, first print the number of the set, as shown in the sample output. Then print the line "`The minimum number of moves is` *k*.", where *k* is the minimum number of bricks that have to be moved in order to make all the stacks the same height.

Output a blank line after each set.

Sample Input

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

Sample Output

```
Set #1
The minimum number of moves is 5.
```

B: Complementary Binary Strings

A binary string is a string composed **only** of the characters '0' and '1'. Two binary strings A and B of equal length are said to be *complementary* if:

- A[i] == '1' whenever B[i] == '0'
  and
- A[i] == '0' whenever B[i] == '1'

(A[i] and B[i] refer to the i^{th} character of A and B respectively) For example, **"01110010"** and **"10001101"** are complementary, and so are **"0001"** and **"1110"**. However, **"10001"** and **"01100"** are not.

Given N binary strings all of the same length, S_1, S_2,…, S_N, you have two find the number of pairs (i, j), where $1 \leq i < j \leq N$, such that the binary strings S_i and S_j are complementary.

Input Specification

The first line of input contains a single integer T (1 <= T <= 10) indicating the number of test cases to follow. For each test case, the first line of input contains a single integer N ($1 \leq N \leq 1{,}000{,}000$) representing the number of binary strings in the first test case. Each of the following N lines contains a binary string. All strings in a single test case will be of the same length, which will not exceed 20 characters.

Output specification

For each test case, output a single integer that represents the number of pairs of binary strings that are complementary. The pairs need not be adjacent. The output from each test case must be on a separate line. In the first test case shown below, the 1st and 2nd strings are complementary, as are the 2nd and 4th strings. In the second test case, the 1st and the 2nd string are complementary, as are the 2nd and 3rd strings.

Sample Input	Sample Output
2 5 0010 1101 1110 0010 0000 4 010 101 010 000	2 2

Authors: Frank Arteaga, Angel Paredes

C: Counting in Base Seven

The ancient race of Septonians who lived in Hawaii near the *Seven Sacred Pools* on the Island of Maui used to keep track of their coconuts by counting in base 7. We know this because of ancient Hawaiian petroglyphs carved into the rock in this area. As an archaeologist, you need to understand counting in 7 if you are to understand these people.

The Problem

Given two 7-digit base-7 numbers x and y, where 0000000 <= x, y <= 1000000, your job is to calculate and print all values in from x to y, inclusive.

Input Specification

There will be only a single test case containing two integers x and y, separated by a single space on the same input line. The values x and y represent the beginning and ending values of the range to be generated. Each number will be exactly 7 digits long, including its leading zeros.

Output Specification

Each 7-digit value from x to y must appear on a separate line of output. Each number must contain 7 digits.

Sample Input

0000356 0000400

Sample Output

0000356
0000360
0000361
0000362
0000363
0000364
0000365
0000366
0000400

D: Dog and Gopher

A large field has a gopher and a dog. The dog wants to eat the gopher, while the gopher wants to run to safety through one of several gopher holes dug in the surface of the field. Neither the dog nor the gopher is a math major; however, neither is entirely stupid. The gopher decides on a particular gopher hole and heads for that hole in a straight line at a fixed speed. The dog, which is very good at reading body language, anticipates which hole the gopher has chosen, and heads at double the speed of the gopher to the hole, where it intends to gobble up the gopher. If the dog reaches the hole first, the gopher gets gobbled; otherwise, the gopher escapes. You have to select a hole for the gopher through which it can escape, if such a hole exists.

Input

The first line of the input file indicates the number of test cases. For each test case, the first line contains one integer and four floating point numbers. The integer **n** denotes how many holes are in the set and the four floating point numbers denote the **(x,y)** coordinates of the gopher followed by the **(x,y)** coordinates of the dog. Subsequent **n** lines of input each contain two floating point numbers: the **(x, y)** coordinates of a gopher hole. All distances are in meters; to the nearest **mm**. There is a blank line between two consecutive test cases.

Output

Your should output a single line for each test case. For each case, if the gopher can escape the output line should read **"The gopher can escape through the hole at (*x*,*y*)."** identifying the appropriate hole to the nearest mm. Otherwise the output line should read **"The gopher cannot escape."** If the gopher may escape through more than one hole, report the one that appears first in the input. There are not more than **1000** gopher holes in a set of input and all coordinates are between **-10000** and **+10000**. All floating-point values are formatted to 3 decimal places.

Sample Input

```
2
1 1.000 1.000 2.000 2.000
1.500 1.500

2 2.000 2.000 1.000 1.000
1.500 1.500
2.500 2.500
```

Sample Output

```
The gopher cannot escape.
The gopher can escape through the hole at (2.500,2.500).
```

Source: Waterloo ACM Programming Contest

E: Freckles

In an episode of the Dick Van Dyke show, little Richie connects the freckles on his Dad's back to form a picture of the Liberty Bell. Alas, one of the freckles turns out to be a scar, so his Ripley's engagement falls through.

Consider Dick's back to be a plane with freckles at various (x,y) locations. Your job is to tell Richie how to connect the dots so as to minimize the amount of ink used. Richie connects the dots by drawing straight lines between pairs, possibly lifting the pen between lines. When Richie is done there must be a sequence of connected lines from any freckle to any other freckle.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line contains $0 < n <= 100$, the number of freckles on Dick's back. For each freckle, a line follows; each following line contains two real numbers indicating the (x,y) coordinates of the freckle.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Your program prints a single real number (rounded to two decimal places) indicating the minimum total length of ink lines that can connect all the freckles.

Sample Input

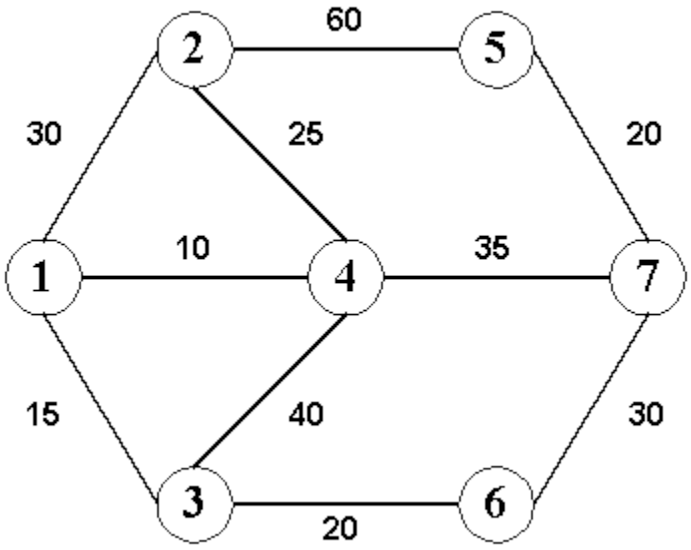
```
1

3
1.0 1.0
2.0 2.0
2.0 4.0
```

Sample Output

```
3.41
```

F: The Tourist Guide

Mr. G. works as a tourist guide. His current assignment is to take some tourists from one city to another. Some two-way roads connect the cities. For each pair of neighboring cities there is a bus service that runs only between those two cities and uses the road that directly connects them. Each bus service has a limit on the maximum number of passengers it can carry. Mr. G. has a map showing the cities and the roads connecting them. He also has the information regarding each bus service. He understands that it may not always be possible for him to take all the tourists to the destination city in a single trip. For example, consider the following road map of 7 cities. The edges connecting the cities represent the roads and the number written on each edge indicates the passenger limit of the bus service that runs on that road.



Now, if he wants to take 99 tourists from city 1 to city 7, he will require at least 5 trips, since he has to ride the bus with each group, and the route he should take is : 1 - 2 - 4 - 7. But Mr. G. finds it difficult to find the best route all by himself so that he may be able to take all the tourists to the destination city in minimum number of trips. So, he seeks your help.

Input

The input will contain one or more test cases. The first line of each test case will contain two integers: N (N <= 100) and R representing respectively the number of cities and the number of road segments. Then R lines will follow each containing three integers: C_1, C_2 and P. C_1 and C_2 are the city numbers and P (P > 1) is the limit on the maximum number of passengers to be carried by the bus service between the two cities. City numbers are positive integers ranging from 1 to N. The (R + 1)-th line will contain three integers: S, D and T representing respectively the starting city, the destination city and the number of tourists to be guided. The input will end with two zeroes for N and R.

Output

For each test case in the input first output the scenario number. Then output the minimum number of trips required for this case on a separate line. Print a blank line after the output of each test case.

Sample Input

```
7 10
1  2  30
1  3  15
1  4  10
2  4  25
2  5  60
3  4  40
3  6  20
4  7  35
5  7  20
6  7  30
1  7  99
0  0
```

Sample Output

```
Scenario #1
Minimum Number of Trips = 5
```

G: Word Counting

Problem Description

You've been tasked with building a major component of a word processing program. Given a line and a word, you are to determine the number of occurrences of that word in the line. Note that the word can be contained within another word in the same line, and these should count as two separate occurrences. Two words are considered equal if they contain the same characters in the same order, ignoring uppercase/lowercase differences.

Input

The first line will contain a number *n* which will specify the number of test cases that will follow. Each test case will be 2 lines, the first line will contain the word that we want to find the occurrences of and the line following that will be the sentence where we want to count these occurrences. Words will be no longer than 30 characters and sentences will not be longer than 600 characters.

Output

Each line of output should be on a separate line and should follow the format:

Line #*n*: Word *word* occurred *x* times

where *n* is the number of the current line/test case, *word* is the word that you are counting, and *x* is the number of occurrences of that word in the sentence. The word "times" always ends with an "s".

Sample Input

```
2
the
the problem is too easy
hArd
This problem isn't that hard. I wonder if they get any harder...
```

Sample Output

```
Line #1: Word the occurred 1 times
Line #2: Word hArd occurred 2 times
```