The 2003 ACM ICPC ASIA Programming Contest Dhaka Site.

Sponsored by IBM
Hosted by BUET

12-13th November 2003

You get 19 Pages
10 Problems

Problem A
Palindrome Numbers

Input: Standard Input
Output: Standard Output

A palindrome is a word, number, or phrase that reads the same forwards as backwards. For example, the name "anna" is a palindrome. Numbers can also be palindromes (e.g. **151** or **753357**). Additionally numbers can of course be ordered in size. The first few palindrome numbers are: **1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, ...**

The number **10** is not a palindrome (even though you could write it as **010**) but a zero as leading digit is not allowed.

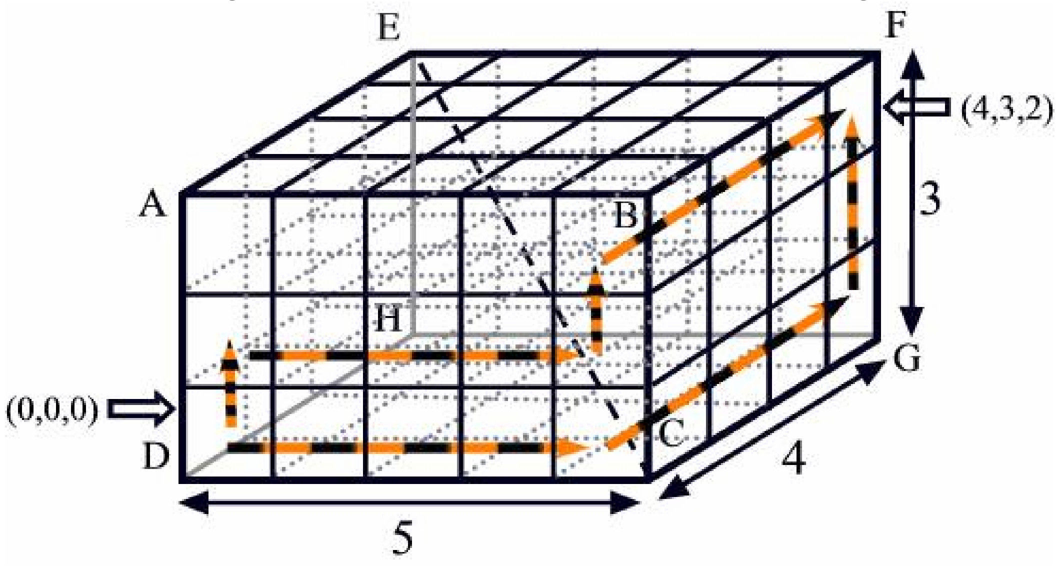
Input
The input consists of a series of lines with each line containing one integer value **i** ($1 \le i \le 2*10^9$). This integer value **i** indicates the index of the palindrome number that is to be written to the output, where index **1** stands for the first palindrome number **(1)**, index **2** stands for the second palindrome number **(2)** and so on. The input is terminated by a line containing **0**.

Output
For each line of input (except the last one) exactly one line of output containing a single (decimal) integer value is to be produced. For each input value **i** the **i**-th palindrome number is to be written to the output.

Sample Input	Output for Sample Input
1 12 24 0	1 33 151

Problem B
Mazes in Higher Dimensions
Input: Standard Input
Output: Standard Output

Mazes in higher dimensions are quite different than mazes in lower dimensions such as two and three. Generally, mazes in lower dimensions have a lot of barriers or walls to confuse the peoples who enter it. But when the dimensions of maze is greater than **3**, mere mortals struggle simply to find their directions so intense barriers are not required. In this problem we will try to study some **N**-dimensional grid mazes and find possible ways of going from one place to another. Our grid maze in three dimension looks like the following:



This three dimensional maze has dimension of **(5 x 4 x 3)** which is made of **60** 3D-blocks. The rules of the maze are:

a) In an **n** dimensional maze one can go towards maximum **2*n** different directions from one place if there is no barrier.
b) The movements are only possible parallel to the axis.
c) The direction of movements should be such that it does not increase distance with the destination. By distance I mean the Cartesian distance.
d) There may be some opaque hyper-blocks in the maze. These blocks do not allow anyone to go through.
e) Total number of hyper-blocks in the maze is not more than **3000000**
f) The maximum dimension length on any axis direction of the maze is **50**.
g) The maximum possible dimension of the maze is **7**.
h) In an **n** dimensional maze let the length along **n** - axes be l_1, l_2, l_3,…, l_n respectively. Then the starting block is always **(0, 0, …, 0)** and the destination is always **(l_1-1, l_2-1, l_3-1, .. , l_n-1)**. This **n**-Dimensional maze is made of **(l_1 x l_2 x l_3 x…x l_n)** **n-**dimensional hyper-blocks. In the **3**-dimensional maze shown above the starting block is **D (0,0,0)** and the destination block is **F (4,3,2)**. The figure above shows two possible ways of going from block **D (0,0,0)** to block **F (4,3,2)**.
i) When the maze makers place opaque hyper-blocks in the maze they make sure that the number of ways to reach the destination is reduced to less than 2^{63}.

Given the description of a maze your job is to determine the number of ways you can go from its lowest block to the highest block as it is said in point **h**. You must know that in a **n**-dimensional Cartesian coordinate system the coordinate of a point **P** can be written in the form **(p_1, p_2, p_3,…, p_n)** and the coordinate of another point **Q** can be written as **(q_1, q_2, q_3,…, q_n)**. And the distance between these two points is

$$\sqrt{(p_1-q_1)^2+(p_2-q2)^2+(p_3-q_3)^2+\ldots+(p_n-q_n)^2}$$

This distance is also known as the Cartesian distance of two points in **n**-dimension.

Input

The input file contains several sets of inputs. The description of each set is given below:

First line of each set contains two integers **DIM(0<DIM<8)** and **R(0<=R<30000)**. Here **DIM** is the dimension of the maze, **R** is the number of opaque blocks. Next R lines contain **DIM** integers **p_1, p_2, p_3, …, p_{DIM}** which is the coordinate of the destination hyper-block. Note that blocks are not point objects. So by coordinate of a block we refer to the point of the block that is nearest to the origin. That's why in the picture above the coordinate of block **D** is **(0,0,0)**. There is a blank line after each set of input.

Input is terminated by a case where both **DIM** and **R** are zero. This case should not be processed.

Output

For each set of input produce one line of output. This line should contain the serial of the output as shown in the sample output. Then it should contain an integer **N** which indicates in how many ways we can reach the destination. It is guaranteed that this number will be less than **2^{63}**.

Sample Input	Output for Sample Input

3 0
5 5 5
 	
3 1
9 8 7
1 1 1
 	
0 0
	Case 1: 756756
	Case 2: 6318655200

Problem C
Cyber cafe

Input: Standard Input
Output: Standard Output

Association of Cyber cafe Managers **(ACM)** has recently discovered that the cyber cafes have not been placed in proper places of the cities. Some parts of the cities have many of them, while some important parts have none. **ACM** is now planning to setup one cyber cafe in each important part of large cities. Each of the new cyber cafes will have one server connected to several terminals. As the number of cyber cafes is now increasing, **ACM** has bought some new servers. The cyber cafes that will be equipped with new servers will be called grade A cyber cafes, and the rest which contains the old servers will be called grade **B** cyber cafes.

Although the old servers are quite good, there is a chance that people will try to avoid grade **B** cyber cafes and prefer grade **A** ones if they have that opportunity. If someone finds that the nearest cyber cafe is of grade **B** and there are more than one grade **A** cyber cafes within **1** km distance, he/she will never use the grade **B** cyber cafe. As a result, that cyber cafe will not get enough customers to support itself. If there is only one grade **A** cyber cafe near (within one km) a grade **B** cyber cafe then people will still use the grade **B** cyber cafe because the only neighboring grade **A** cyber cafe may be filled with people. So, ACM is planning to build grade **A** and grade **B** cyber cafes in such a way that no grade **B** cyber cafe is within **1** km distance of two or more grade **A** cyber cafes.

Figure 1

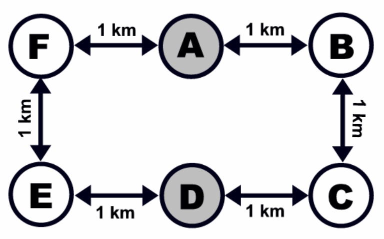


Figure 2

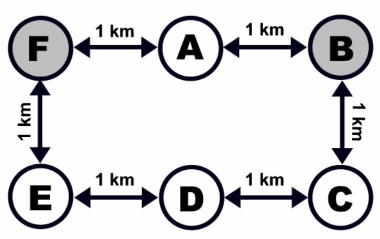


Figure: In the figure above the circles represent cyber cafes. The grey circles indicate cyber cafes with new servers and the white circles indicate cyber cafes with old servers. Figure 1 is a valid configuration as two or more grey circles are not adjascent with one white circle. Figure 2 is not a valid configuration because white circle A has two grey circles (F and B) within 1 km. These pictures are related with the first sample input.

ACM has already rented one building for each cyber cafe. The rented buildings of a city have been marked with single uppercase letters for identification. **ACM** has also decided how many new servers should be allotted to each city. Now, it is time to place the servers in the cyber cafes. Given all the pairs of buildings that are within **1** km distance of each other, your job is to find out the possible ways of placing the servers meeting the above criterion. All the new servers are similar. The old ones are also similar. All cyber cafes should have exactly one server. For examples if there are five cyber cafes and two new servers then three (**5-2=3)** old servers will be required.

Input

Input consists of less than **60** (Sixty) datasets. Each dataset consists of the followings:

- A line containing the name of the city (which has **2** to **16** alphanumeric characters).
- A line containing **3** positive integers **n**, **s** and **p** denoting respectively the number of buildings, the number of new servers and the number of building pairs that are within **1** km from each other **($0 < s < n < 17$)**. The integers are separated by exactly one space. The **n** buildings are marked with **`A`**, **`B`**, etc. in that order.
- Next line contains **p** pairs of uppercase letters. Each pair indicates that the corresponding buildings are within **1** km of each other. One pair is separated from the next with exactly one space.

The end of input is marked with a line consisting of **`"TheEnd"`**.

Output

For each set, print **2** lines. The first line should contain the name of the city. The next line should contain the number of possible ways to place the servers. The last line of output should be a line consisting of the string **`"TheEnd"`** (without the quotes).

Sample Input

```
Dhaka
6 2 7
AB BC CD DE EF FA AD
Chittagong
4 3 4
AB AC AD BC
Sylhet
3 2 3
AB BC CA
TheEnd
```

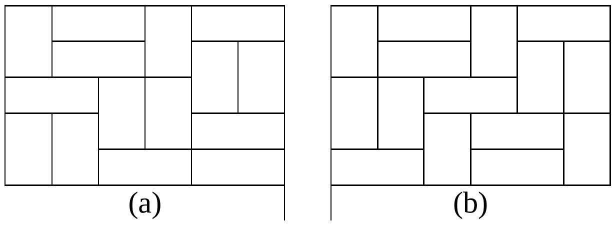
Output for Sample Input

```
Dhaka
9
Chittagong
1
Sylhet
0
TheEnd
```

Problem D
Unbreakable Floor

Input: Standard Input
Output: Standard Output

Alan bought a new house. He likes rectangles, so he wants his floor full of identical rectangular shapes. Imagine he has a floor of **5 x 6**, he may fill this floor with rectangles of **1 x 2** in at least two ways:



Picture **(a)** shows a 'breakable' layout, since there is a straight line through the whole floor which divides the floor into two parts -- a **5 x 4** rectangle and a **5 x 2** rectangle, and all the **1 x 2** rectangles are not destroyed.

Picture **(b)** shows a 'unbreakable' layout, since you cannot divide it into two parts without destroying any **1 x 2** rectangle.

Alan likes unbreakable floorings, but he's not sure if it is possible for any size of floor and rectangle shape. Can you tell him?

Input
The first line contains the number of tests **t(1<=t<=40)**. Each case consists of a single line with four positive integers **p**, **q**, **a**, **b** (**1<=p, q, a, b<=10000**).

Output
For each test case, print the case number first. Then print the word 'Yes' if it is possible to make a unbreakable floor of **a** x **b** with rectangles of **p** x **q**, otherwise print the word 'No'. Answer for each case should be in exactly one line.

Sample Input

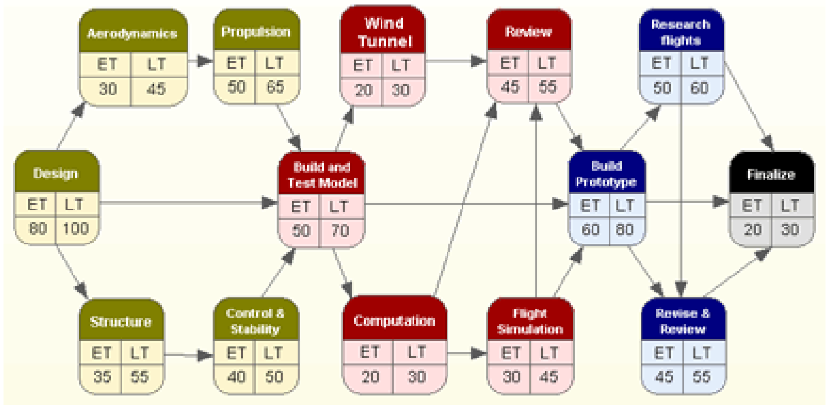
```
3
1 2 5 6
1 2 3 17
2 3 11 18
```

Output for Sample Input

```
Case 1: Yes
Case 2: No
Case 3: Yes
```

Problem E
Find the Project

Input: Standard Input
Output: Standard Output



The job market is not holding a good prospect for Computer Science graduates lately. As a result a lot of **CS** graduates here are either going for **MBA** or looking for a job, which is not quite related to their field of study. Turza, a fresh **CS** graduate is one such example. He with his sound knowledge of **CS** could not find a single job where he could make use of what he knows. He ends up getting a job of project supervision, which is basically keeping track if the parts of the project are being completed according to the schedule. What a waste of four years' hard work!

A guy has to earn his living and we all understand that. But there should also be some sort of satisfaction in the job. That's what Turza is trying to find now. He is desperately searching for something interesting in his work. But the only interesting thing for him is mathematical problems. Fortunately for Turza, he has been able to create and solve one math problem from his not so interesting work. We would like to leave it for you to see if you can do it too. He says:

"The project I supervise has many parts. Some of the parts are dependent on other parts. So if part **X** is dependent on part **Y** and **Z**, we'd have to make sure that both part **Y** and **Z** is finished before we start working on part **X**. For some strange reason which I really don't have any clue to, here we must finish all the parts that are currently not dependent on any other parts. Once all these parts are completed, we proceed on to completing the other parts following the same rule. It is fairly easy to calculate how many ways are there that this entire project can be completed. However, I am more interested in the solution of the inverse problem. Given the number of parts we have in a project and the number of ways that the project can be completed (if we follow the procedure I just explained) can you find the dependency list of the parts in the project? It is quite obvious that the answer need not be unique, so any valid answers would do."

If we want to calculate the number of ways to finish a project the number can get quite big, even if the number of parts in the project is not so large. So here we prefer to use the prime factorized form for the count. You can safely assume that for every count and number of parts we give you, there is at least one project that they correspond to. Your task is simply to find one such project and list the dependency among the parts.

Input

The first line of the input gives you the number of test cases, **T (T <= 20)**. Then you would have the description of **T** test cases. Every test case starts with two integers in one line. The first integer **N (1 <= N <= 50)** is the total number of parts in the project. The next integer is the number of primes, **P (P <= N)** in the prime factorization of the count (the number of ways to complete the project). In the next **P** lines **P** pairs are listed. The first number in each pair is the prime number and the second number is the number of times it occurs in the count. Please note that we only list a prime number if it occurs nonzero times in the count.

Output

For each of the test cases you have to print the serial number of the test case in the format **"Case#x:"** where **x** is the serial number starting from **1**. In the next **N** lines you have to list the dependency list for each of the projects. The **i** th dependency list starts with an integer which gives us the number of parts that dependent on the **i** th part. Then we would have the labels of that many parts on the same line. These labels are integers in the range **1** to **N** and should be separated by a single space. If there is more than one such project any one of them will be accepted. Please note that your project description must not be such as to mean that it is impossible to finish the project. And you must not mention a dependency more than once.

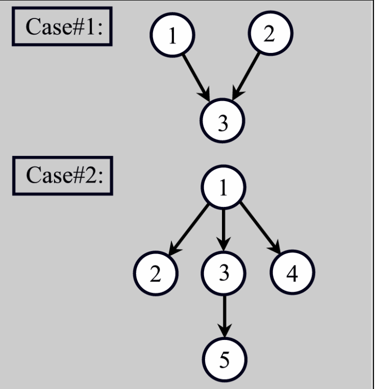
Sample Input

```
2
3 1
2 1
5 2
2 1
3 1
```

Output for Sample Input

```
Case#1:
1 3
1 3
0
Case#2:
3 2 3 4
0
1 5
0
0
```

Illustration



Problem F
Strange Research
Input: Standard Input
Output: Standard Output

Aoccdrnig to a rseearch at an Elingsh uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoatnt tihng is taht the frist and lsat ltteer is at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit any porbelm. Tihs is bcuseae we do not raed ervey lteter by itslef but the wrod as a wlohe.

I hope you did not have much trouble reading the paragraph above and have realized that the research result is somewhat true. If you are still having trouble reading the paragraph above go to the end of this problem to see the actual paragraph.

At first glance this research may appear as a joke but if you think for quite sometime you will find that this is not a joke at all since words of length **1**, **2**, **3** and **4** are merely affected by the scrambling that is being suggested here. And in a normal English text, words of length less than five makes almost **62%** of the total words. However to solve this problem you neither need this statistics nor need to prove the correctness of this research.

SCORPIONS
SCOONIRPS
Original Word
Scrambled Word

In this problem you will be given a dictionary of correct words and list of scrambled words. The first and the last letter of a scrambled word are the same as the original word. Position of letters between them may or may not be altered. Your job is to find out the total number of words in the dictionary that can actually be the correct form of the given scrambled word. You also need to find the lexicographically smallest and largest such word in the dictionary, when the word is present in the dictionary.

Input
The input file contains a single set of input. The first line of the input file is an integer **N** **(0<N<500001)** which indicates how many words are there in the given dictionary. Each of the next **N** lines will contain one word. All the words have less than **9** characters. The very next line contains an integer **Q (Q<50001)** which indicates the total number of query. Each of the next **Q** lines contain **1** word (All query words have less than **16** characters). All words in the input file are made with lowercase alphabets.

Output
For each query you should output a single line. This single line should contain an integer **F** (must) and two strings **S1** and **S2** (These two strings will be present only when **F>0**). The integer **F** denotes how many words are there in the dictionary that can be the correct form of

the scrambled word in the query. **S1** is the lexicographically smallest such word in the dictionary and **S2** is the lexicographically largest such word in the dictionary.

Sample Input

```
5
aabab
aaaab
aaabb
aaaaa
xyxyx
5
aaaaa
aabab
zzzzz
kkkkk
aaabb
```

Output for Sample Input

```
1 aaaaa aaaaa
2 aaabb aabab
0
0
2 aaabb aabab
```

According to a research at an English university, it doesn't matter in what order the letters in a word are, the only important thing is that the first and last letter is at the right place. The rest can be a total mess and you can still read it without any problem. This is because we do not read every letter by itself but the word as a whole.

Problem G
Circle Strafing

Input: Standard Input
Output: Standard Output

A recent election and violent unprecedented secession has firmly established you as warlord of the new nation Illbebakia. Of course, there is still some dissent by your weakling serfs, so you must solidify your military stranglehold on your nation. And naturally, there are neighbors to subdue while you're at it. These tasks are, like so many things, best accomplished by randomly blowing things up. Fortunately, you have a proven talent in this area.

Even as we speak, your warplanes are dropping large bombs all over the country. You need some way to determine the extent of the carnage. If the pilots have served you well, they may live for another precious day. You don't really care about the property damage or the massive casualties; it's simply the psychological effect (shock and woe) that's important. As such, all you want to know is the total area of devastation.

Every bomb has a destruction radius. Anything within that radius is completely eradicated. Computing the area for one bomb is fairly simple, but for many it isn't quite so easy. However, your new nation has a surprisingly large proportion of skilled programmers, so you have respectfully requested their assistance. The survivors of this request (ie, those who cooperated) are now hard at work, writing a program to solve this problem...

Fig 1:A typical explosion

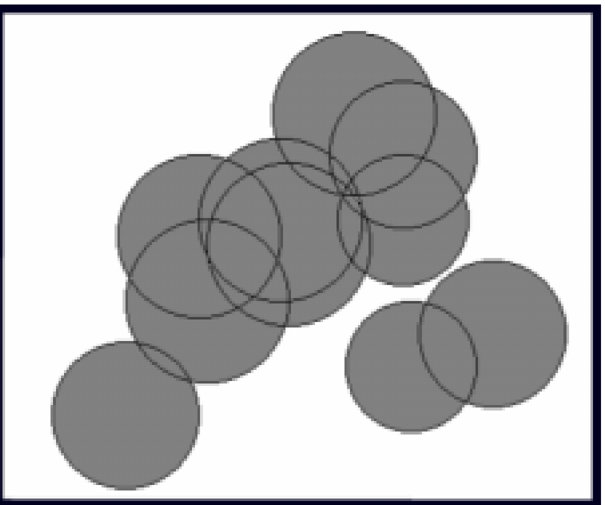


Fig 2: Computing the area for one bomb is fairly simple, but for many it isn't quite so easy. The area is shown in grey.

Input
Input consists of a number of cases. Each case lists all of the bombs dropped on one day of your rule. The first line of case contains **n**, the number of bombs. The next **n** lines each contain the **x** and **y** coordinates where one bomb exploded, and its destruction radius. There

will be at most 100 bombs. Coordinates given are real numbers between **0** and **100**, and the radius is a real number between **0** and **10**.

There will be at most **50** cases. The last day of bombing will be followed by a line containing **0**. This case must not be processed.

Output

For each case, output the area of destruction from all of the bombs from that day, accurate to three decimal places.

Sample Input

```
1
0 0 10
2
0 0 10
0 10 10
0
```

Output for Sample Input

```
314.159
505.482
```

Problem H
Prefix Codes

Input: Standard Input
Output: Standard Output

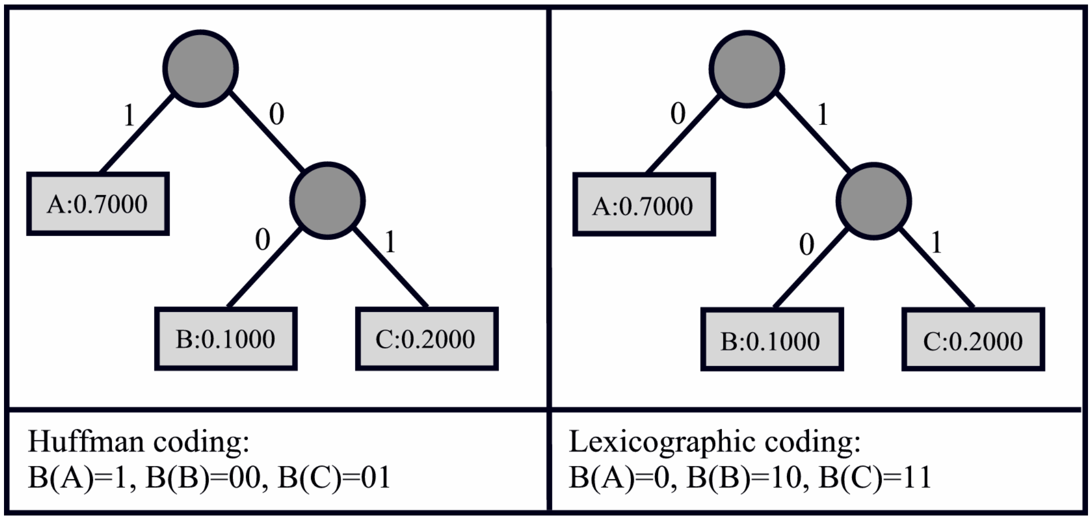
Given an alphabet **S**, and a probability **Prob(a)** for each $a \in S$, a binary prefix code represents each ***a*** in *S* as a bit string **B(*a*)**, such that **B(*a₁*)** is not a prefix of **B(*a₂*)** for any $a_1 \neq a_2$ in S.

Huffman's algorithm constructs a binary prefix code by pairing the two least probable elements of **S**, a_0 and a_1. a_0 and a_1 are given codes with a common (as yet to be determined) prefix **p** and differ only in their last bit: $B(a_0) = p_0$ while $B(a_1) = p_1$. a_0 and a_1 are removed from **S** and replaced by a new element **b** with $\mathrm{Prob(b)} = \mathrm{Prob}(a_0) + \mathrm{Prob}(a_1)$. **b** is an imaginary element standing for both a_0 and a_1. The Huffman code is computed for this reduced **S**, and **p** is set equal to **B(b)**. This reduction of the problem continues until **S** contains one element ***a*** represented by the empty string; that is, when **S = {*a*}**, **B(*a*) = ε**.

Huffman's code is optimal in that there is no other prefix code with a shorter average length defined as

$$\sum_{a \in S} \mathrm{Prob}(a) \times |B(a)|$$

One problem with Huffman codes is that they don't necessarily preserve any ordering that the elements may have. For example, suppose **S = {A, B, C}** and **Prob(A) = 0.7**, **Prob(B) = 0.1**, **Prob(C) = 0.2**. A Huffman code for **S** is **B(A) = 1**, **B(B) = 00**, **B(C) = 01**. The lexicographic ordering of these strings is **B(B)**, **B(C)**, **B(A) [i.e. 00,01,1]**, so the coding does not preserve the original order **A**, **B**, **C**. Therefore, algorithms like binary search might not work as expected on Huffman-coded data.



Given an ordered set **S** and **Prob**, you are to compute an ordered prefix code - one whose lexicographic order preserves the order of **S**.

Input

Input consists of several data sets. Each set begins with **$0 < n \le 100$**, the number of elements in **S**. *n* lines follow; the **i-th** line gives the probability of **a_i**, the **i-th** element of **S**. Each probability is given as **0.dddd** (that is, with exactly four decimal digits). The probabilities sum to **1.0000** exactly. A line containing **0** follows the last data set.

Output

For each data set, compute an optimal ordered binary prefix code for **S**. The output should consist of one line giving the average code length, followed by **n** lines, with the **i-th** line giving the code for the **i-th** element of **S**. If you have solved the problem, these **n** lines will be in lexicographic order. If there are many optimal solutions, choose any one.

Output an empty line between cases.

Sample Input

```
3
0.7000
0.1000
0.2000
3
0.7000
0.2000
0.1000
0
```

Output for Sample Input

```
1.3000
0
10
11

1.3000
0
10
11
```

Problem I
Highway Monitor

Input: Standard Input
Output: Standard Output

The population of Mars has increased rapidly and so has the traffic on its highways. This not only increased traffic jams but also made it difficult to locate outlaws in rush hours. So, the traffic authority has decided to establish a monitoring system on the highway. The idea is like this – they will setup some traffic cameras at some strategic points. These cameras will constantly monitor the traffic passing by (in both directions) and record the video footage for future analysis. The scientists working at Mars are pretty smart. They have a large number of cities and they designed the highways as segments between these cities (and nothing else). What this means is that each highway segment starts and finishes at exactly two different city junctions. No two highways meet anywhere other than at a city junction. Also note that all highway segments allow two-way traffic. In order to avoid theft of cameras, the traffic authority wants to setup cameras only at these city junctions. The cameras can take pictures from all directions simultaneously. So a camera at a city junction can monitor all the highway segments ending at that junction.

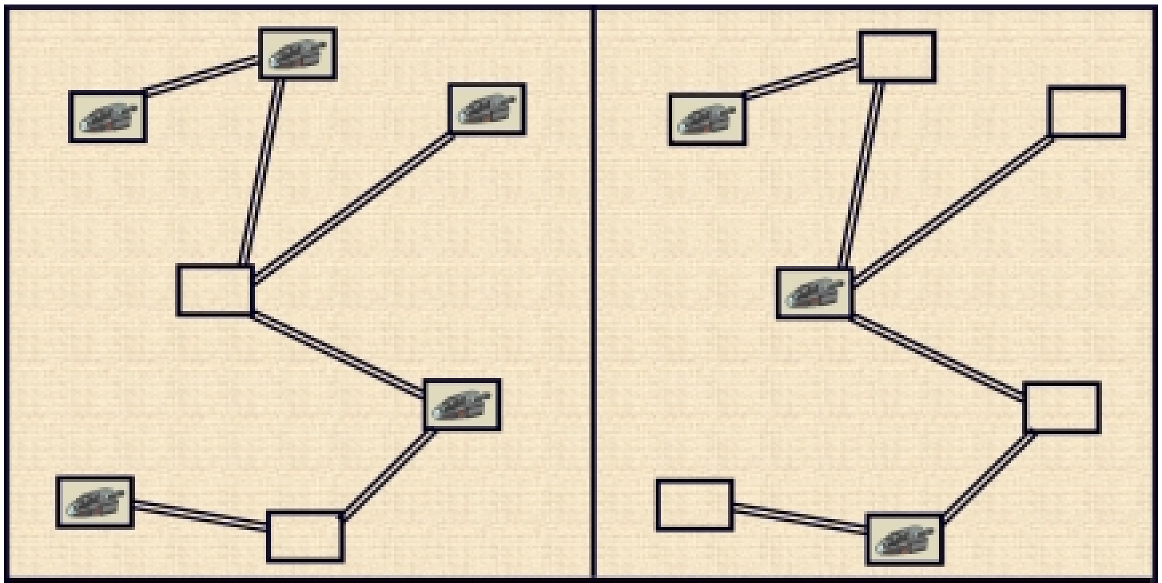


Figure: In the picture on the left we can see that all the streets are being monitored by five cameras. In the picture on the right the same street configuration is being monitored by only three cameras.

Imagine yourself as a scientist in Mars. You are given a certain number of cameras and a description of all highway segments you need to monitor. You are to determine whether it is possible to monitor all the highway segments using the given number of cameras. Moreover, if it is possible indeed, you are to identify the city junctions where you may place the cameras.

Input

The input file will contain multiple test cases. First line of input contains a single integer that specifies how many test cases you have in the input (**2** in sample input). Actual test data starts from the next line.

First line of test data contains three integers: **N**, **1 <= N <= 1000**, **H** and **k**, **1 <= K <= 18**. Here **N** is the number of city junctions, **H** is the number of highway segments among them and **K** is the number of cameras. In the first test case of sample input we have **N = 5**, **H = 7** and **K = 2**. Next **H** lines describe the highway segments, one segment in each line. Each highway segment is described by two integers **x** and **y** which specify that we have a (two-way) highway segment between city junctions **x** and **y**. Next test case begins at the end of this highway segment list. There can be at most 15 test cases.

(A cautionary note: The number of city junctions can be as high as 1000)

Output

For each test case, first you have to print the test case number as shown in the sample output. Then you have to print **"yes"** or **"no"** depending on whether you can monitor all the described highway segments with at most the given number of cameras. Moreover, if your answer is **"yes"**, you have to print a list of city junctions (in any order) where you will setup the cameras. Note that there can be multiple such lists. In this case you may print any such list that satisfies the requirement. The entire output corresponding to a test case must be in a single line.

Sample Input

```
2
5 7 2
1 2
1 3
1 5
2 4
2 5
3 5
4 5
5 7 3
1 2
1 3
1 5
2 4
2 5
3 5
4 5
```

Output for Sample Input

```
Case #1: no
Case #2: yes 1 2 5
```

Problem J
Volume Measurement

Input: Standard Input
Output: Standard Output

Please look at the picture below: A solid cube is placed within a pyramid shaped frame. The base of the pyramid frame is square **ABCD**. The length and width of this base is **l,** (The image below shows **l** and **w**. Assume that **w** equals **l**) and the height of the pyramid frame is **h**. The length of one (or all) side of the solid cube is **a.** Point **A** is considered the origin, x-axis is along **AX** and y-axis is along **AY**. So the coordinates of **A**, **B**, **C**, **D** are **(0, 0)**, **(l, 0)**, **(l, l)** and **(0, l)** respectively. The peak of the pyramid frame is **E** which is straight above the center of the base **ABCD**. The center of the base of the cube is **O(l/2,l/2)** which is also the center of the base of the pyramid and **FG** makes an angle **θ(0<θ<90)** with **x**-axis. Surely for many values of **a** or **θ** the cube will not fit within the pyramid frame. Your job is to detect situations whether the cube fits within the pyramid and if it does fit then find the volume of the combined object.

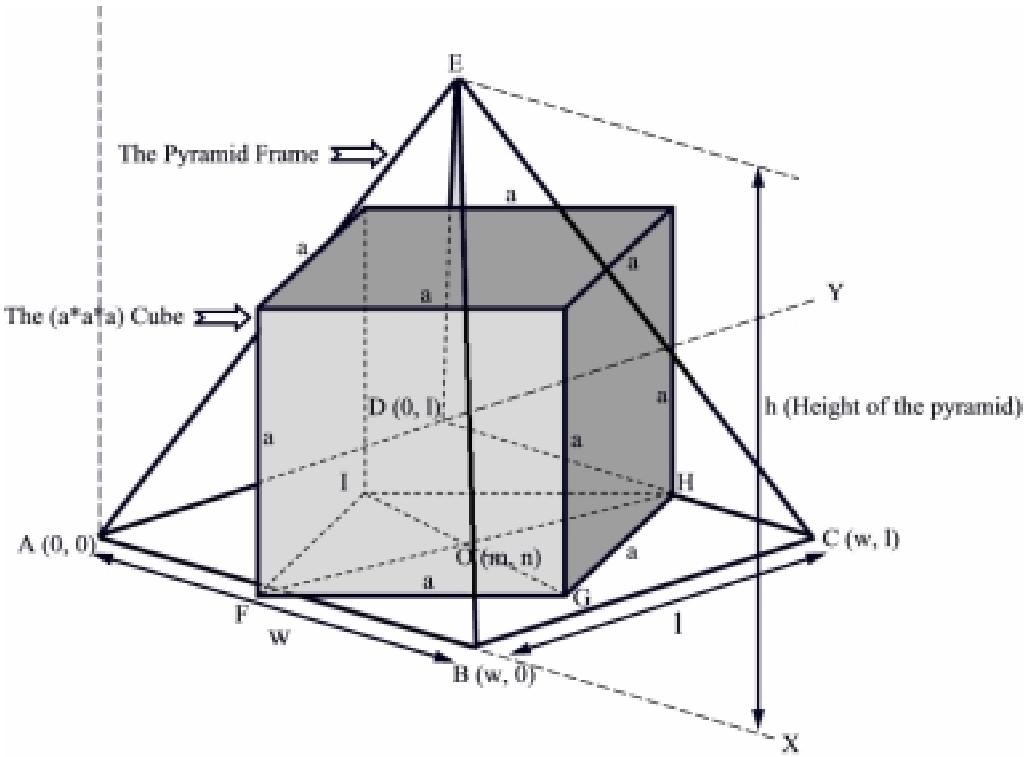


Figure: A square placed within a pyramid frame. Please note that the image above is not exact with the problem statement. In our problem the length and width of the base of the pyramid are equal **(l=w)**. And in our problem the square is placed right at the center of the base of the pyramid. That is not the case in the figure above.

Input

The input file contains several sets of input. The first line of the input file contains an integer **N** that indicates how many sets of input are there. Each of the next **N** lines contains a single set of input. All numbers in the input are less than **200**.

Each set contains four positive integers **l, h, a, θ**.

There will be no such inputs which will cause floating-point errors to produce different outputs.

Output

For each set of input you must produce two lines of output. The first line should contain the serial of the output as shown in the output for sample input. If for the given input it is impossible to fit the cube within the pyramid shaped frame (Some parts of the cube may be outside the pyramid keeping the shape of the frame intact) print in the second line **"IMPOSSIBLE."**. Otherwise in the second line print the volume of the combined object. This volume should have three digits after the decimal point.

Sample Input	Output for Sample Input

2
10 10 10 45
10 10 1 50
	Case 1:
	IMPOSSIBLE.
	Case 2:
	333.333