ACM-ICPC Asia Phuket
Regional Programing Contest 2009

Sponsored by IBM

Hosted by Prince of Songkla University, Phuket Campus

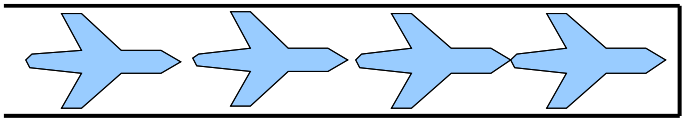
4th November 2009

- There are **11 problems** (A-K) to solve within **300 minutes** (5 hours).
- Solve as many problems as you can, in an order of your choice.
- Use **C** or **C++** or **Java** to program at your convenience for any problems.
- Input and output of each program are **standard input** and **output.**

Platinum Local Sponsor

Problem A
Airplane Parking

During this economic crisis time, Jack has started an incredible new business related to air travel, a parking-lot for airplane. He bought a very large land to park airplanes. However the land is very narrow, so that the only way airplanes can go in or go out of the parking lot must be in the **Last-In First-Out** fashion (see picture below). He only has spaces in the parking lot so he **cannot** take some airplane at the end out so that other airplanes can move.



Because of the limitation of the parking lot, it is not possible to accommodate all requests for parking. Each request consists of the planned arrival time and planned departure time, which are the times the airplane arrives at the parking lot. An example below shows a request table for 4 planes.

Airplane	Arrival	Departure
1	1	10
2	2	5
3	3	7
4	6	9

In this case, it is possible to accommodate airplane 1, 2, and 4. But it is not possible to accommodate both airplanes 2 and 3.

It is possible that different planes plan to arrive or depart the parking lot at the same time. Jack has the best crews working with him, so that they will manage to arrange the plane to the parking lot in the best way that if it is possible to park and take out the planes they will be able to do it. Consider another example.

Airplane	Arrival	Departure
5	10	12
6	10	15
7	13	17

Although airplane 5 and 6 arrive at the same time, Jack's crews know that airplane 5 will have to be out before airplane 6, so when both airplanes arrive they put airplane 6 in first, following by airplane 5.

Given a list of parking requests, you want to find the maximum number of airplanes that can be parked in this parking lot, provided that they can only depart in the Last-In First-Out fashion.

A

Input

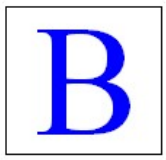
The first line contains an integer T, the number of test cases ($1 \leq T \leq 5$). Each test case is in the following format.

The first line starts with an integer N ($1 \leq N \leq 300$) denoting the number of airplanes. The next N lines describe the request table. Each line $1 + i$, for $1 \leq i \leq N$, contains two integer S_i and T_i, ($0 \leq S_i < T_i \leq 1{,}000{,}000{,}000$) which are the planned arrival time and planned departing time for airplane i.

Output

For each test case, you program must output a single line consisting of one integer, the maximum number of airplanes that can be parked in Jack's parking lot.

Sample Input	Sample Output
2 4 1 10 2 5 3 7 6 9 3 10 12 10 15 13 17	3 2



Problem B
Elias Omega Coding

Introduction

Elias code can be used to efficiently encode positive integers when there is no prior information about the cardinality of the integers to be encoded and it is known that the probability of getting a large integer is smaller than or equal to the probability of getting a small integer. **For practical reasons, however, in this contest we do pose a limit on the size of the input integers. For the same reason we restrict the input integer to be greater than 1.**

Elias developed three variants of the code: the Elias gamma, Elias delta, and Elias omega coding methods. This problem presents the Elias gamma and Elias omega methods and calls for implementing the Elias omega code. The following is a background, definition, and illustration of the problem.

Background

Suppose that Alice wants to transmit a positive integer n to Bob through a binary channel and let β n stand for the binary representation of n. If Bob knows |β(n) | (the number of bits required for the binary representation of n) in advance, then Alice should use β(n) for the transmission. On the other hand, if Bob does not have this information, then Alice can first send |β(n)|, using efficient and distinguishable encoding, then she can send the actual beta code F_1= β (n). The end result is a two field code < F_1,F_2> .

Elias code and its variants differ in the way they encode these two pieces of information (F_1 and F_2). The main difference between variants lies in the representation of F_1 . This may imply modifications in the representation of F_1 . In addition, some of the variants apply repetition or recursion to the representation of F_1. **We use a specific variant specified below.**

Definition

Formally, in Elias gamma coding, a positive integer n is encoded using two concatenated bit fields. The first field, the prefix, contains [$\log_2$ n] bits of 0 ([x] is the floor of x). The second field, the postfix, is the actual binary representation of n using [$\log_2$ n]+1 bits. For example, the binary representation of the decimal number 9 is 1001. Under Elias coding 9 is encoded as 0001001. The first three leading zeros denote that four bits are required for the binary representation of 9. The next four bits contain the binary representation of 9. Elias delta code applies the gamma code to the prefix([$\log_2$ n]) of the gamma code and *Elias omega code* applies a recursion over the prefix Elias gamma representation of [$\log_2$ n].

Illustration (detailed example)

To further illustrate the Elias omega code, consider the integer 536870907. The binary representation of this integer is 1 1111 1111 1111 1111 1111 1111 1011 which is required 29 bits. Hence, in the first step it is encoded as follows:

< F_1,F_2> =

< 0000 0000 0000 0000 0000 0000 0000, 1 1111 1111 1111 1111 1111 1111 1011>

where blanks, dots, commas, and brackets, are inserted for readability.

To emphasize, for this contest the recursion stops when the first field of a recursive stage contains one 0.

Looking at all the bits generated by all the steps and using β(n) to denote the binary representation of n, we have:

a. <28 0's, β(536870907)> =
< 0000 0000 0000 0000 0000 0000 0000, 1 1111 1111 1111 1111 1111 1111 1011>
b. < <4 0's, β(28)>, β(536870907) > =
< <0000, 1 1100>, 1 1111 1111 1111 1111 1111 1111 1011>
c. <<<2 0's, β(4)>, β(28)> , β(536870907) > =
< <<00, 100>, 1 1100>, 1 1111 1111 1111 1111 1111 1111 1011>
d. <<<<one 0, β(2)>, β(4)>, β(28)>, β(536870907) > =
< <<<0, 10>, 100>, 1 1100>, 1 1111 1111 1111 1111 1111 1111 1011>

Taking away all the blanks, dots, commas, and brackets we get that the Elias omega code of 536870907 is: 01010011100111111111111111111111111111011. This code is distinguishable (uniquely decodable). It can be decoded in a unique way and yield back the number 536870907.

Problem statement

Given some positive integers in the range of 2 – $2x10^9$, you are to write a program to produce the Elias omega codes for these integers.

Input

The input contains positive integers each in a separate line. Each integer is in between 2 - 2,000,000,000, inclusive. The end of input is indicated by a zero. The input can contain up to one hundred lines.

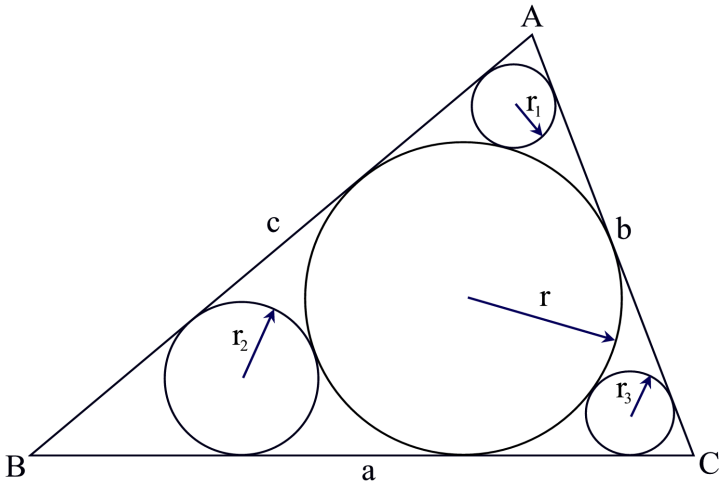
Output

The output consists of lines each corresponding to an input integer except the last zero. Each line contains the Elias omega code of each input integer. The output should not contain any blanks or blank lines.

Sample Input	Sample Output
2 510 7 120000 536870905 49 5 0	010 0111000111111110 010111 01010010000111010100110000000 01010011100111111111111111111111111111001 010101110001 010101

Problem C
In-circles Again

In the figure below you can see triangle ABC and its in-circle (Circle that touches all the sides of a triangle internally). The radius of this in circle is r. Three other circles are drawn. Each of them touches two sides of this triangle and the in circle of ABC. The radiuses of these circles are r1, r2 and r3.



Given the values of r, r1, r2 and r3 you will have to find the area of triangle ABC.

Input

The input file can contain up to 1000 lines of inputs. Each line contains four positive floating-point numbers which denotes the values of r, r1, r2 and r3 respectively.

Input is terminated by a line containing four negative integers.

Output

For each line of input produce one line of output. This line contains serial of output followed by a floating-point number which denotes the area of triangle ABC. This floating-point number may have two digits after the decimal point. You can assume that for the given values of r, r1, r2 and r3 it will always be possible to construct a triangle ABC. If required you can assume that pi = 3.141592653589793 and also use double precision floating-point numbers for floating-point calculations. You can assume that there will be no such input for which small precision errors will cause difference in printed output. Look at the output for sample input for details.

Sample Input	Sample Output
49.1958415692 5.3025839959 20.7869367050 31.8019699761 186.6830516757 71.9474500429 84.8796672233 37.6219288070 -1 -1 -1 -1	Case 1: 18237.14 Case 2: 195777.32

Problem D
Rating Hazard

A very important aspect of web portals is customer reviews. The customers can rate any product in the web portal. Generally, a customer can rate a product from one star to five star. Based on the rating of all the customers the average customer rating for a product is shown. Look at the figure on the left to get a clear idea. For example if three customers rate a product as 3 star, 4 star and 4 star respectively then the average rating will be (3+4+4)/3 = 3.67 (Rounded to two digits after the decimal point). In the figure on the left 847 customers have rated a product and 597, 189, 26, 11 and 24 customers have rated the product as 5 star, 4 star, 3 star, 2 star and 1 star respectively. So the average rating is: $\frac{597\times5+189\times4+26\times3+11\times2+24\times1}{847}=4.56316411$ (Rounded to eight digits after the decimal point).

Most web portals display the total number of people who have rated the product (As more people rates the product the more reliable the rating is) but do not display the numeric value of the average rating. In the web portal of warzone (A renowned web portal) the total number of customers that have rated a product (In the figure above the total 847 customers have rated the product) and the average rating is stored in two different tables. The average rating is stored, rounded to n (0<n<9) digits after the decimal point so its value is not always the exact average value. Unfortunately, the table that stored the total number of people that rated different products somehow got corrupted. All information available in the database now is the average rating (rounded to at most eight digits after the decimal point). They do not want to lose the huge number of customer ratings they have received throughout 10-15 years but also they cannot cheat with their customers by guessing the number of raters or voters. So from the average rating they want to determine the minimum possible number of people that rated that product. You have to help them find it out by writing a program.

Input

The input file can contain up to 2000 lines of inputs. Each line contains a non-negative floating-point number **v (1 ≤ v ≤ 5)**. This number will have minimum one digit and maximum eight digits after the decimal point. If this number has n digits after the decimal point then you have to assume that the value of the average is given rounded to n digits after the decimal point.

Input is terminated by a line containing a negative number.

Output

For each line of input produce one line of output. This line contains the serial of output followed by an integer T which denotes the minimum number of voter that is required for this average rating.

Sample Input	Sample Output
1.15 4.56316411 4.56316 3.67 3.66 -1.00	Case 1: 13 Case 2: 847 Case 3: 190 Case 4: 3 Case 5: 29

Problem E
Relational Operators

The relational operators in C programming language are shown in the table below. Of course the last two are also known as equality operators.

No	Operator	Meaning
1	>	Greater Than
2	>=	Greater Than or Equal to
3	<	Less than
4	<=	Less than or Equal
5	==	Equal
6	!=	Not Equal

These operators compare the value of the two operands it is working on (The value on its left and the value on its right) and returns true or false (one or zero). For example value of `2 > 3` is interpreted as "false" (As 2 is actually less than 3), value of `3 != 4` is interpreted as true and value of 3 >= 3 is also interpreted as true. You have to find out this interpretation using a program.

Input

The input file contains around 12000 line of input. Each line contains two integers a and b separated by an operator "`>`", "`>=`", "`<`", "`<=`", "`==`" or "`!=`". Input is terminated by a line which contains an "E" instead of the operators. Note that there is also a space between any operator and operand. You can assume (-10000 ≤ a, b ≤ 10000).

Output

For each line of input produce one line of output. This line contains the serial of output followed by a string "true" or "false" (without the quotes) which denotes how the expression is interpreted in C language. Look at the output for sample input for details.

Sample Input	Sample Output

`3 != 3`
`4 < 4`
`4 <= 5`
`3 E 3`
	`Case 1: false`
	`Case 2: false`
	`Case 3: true`

Problem F
Your Ways

You live in a small well-planned rectangular town in Phuket. The size of the central area of the town is *H* kilometers x *W* kilometers. The central area is divided into *HW* unit blocks, each of size 1 x 1 km^2. There are *H* + 1 streets going in the West to East direction, and there are W + 1 avenue going in the North-South direction. The central area can be seen as a rectangle on the plane, as shown below.

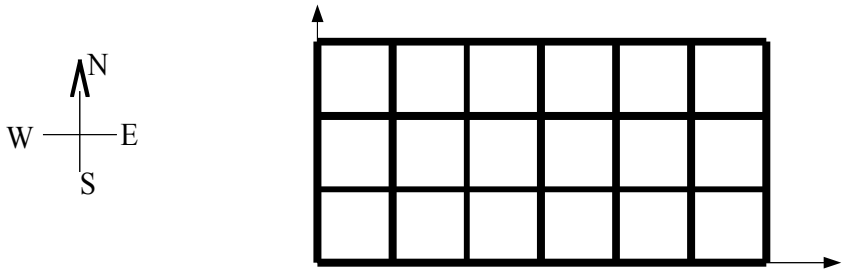


Figure 1. The central area for a town where H = 3, and W = 6.

We can identify each intersection by its co-ordinate on the plane. For example, on the Figure above the bottom-left corner is intersection (0,0), and the top-right corner is intersection (6,3).

Your house is at the bottom-left corner (i.e., intersection (0,0)) and you want to go to the university at the top-right corner (i.e., intersection (*W*,*H*)). More over, you only want to go to the university with wasting any efforts; therefore, you **only want to walk from West-to-East and South-to-North directions**. Walking this way, in the example above there are 84 ways to reach the university.

You want to go to the university for *K* days. Things get more complicated when each morning, the city blocks parts of streets and avenues to do some cleaning. The blocking is done in such a way that it is **not** possible to reach parts of the streets or avenues which is blocked from some other part which is blocked as well through any paths containing **only** West-to-East and South-to-North walks.

You still want to go to the university using **the same West-to-East and South-to-North strategy**. You want to find out for each day, *how many ways* you can reach the university by only walking West-to-East and South-to-North. Since the number can be very big, we only want the result **modulo 2552**.

Input

The first line contains an integer T, the number of test cases ($1 \le T \le 5$). Each test case is in the following format.

The first line of each test case contains 3 integers: W, H, and K ($1 \le W \le 1{,}000$; $1 \le H \le 1{,}000$; $1 \le K \le 10{,}000$). W and H specify the size of the central area. K denotes the number of days you want to go to the university.

The next K lines describe the information on broken parts of streets and avenues. More specifically, line 1 + i, for $1 \le i \le K$, starts with an integer Q_i ($1 \le Q_i \le 100$) denoting the number of parts which are blocked. Then Q_i sets of 4 integers describing the blocked parts follow. Each part is described with 4 integers, A, B, C, and D ($0 \le A \le C \le W$; $0 \le B \le D \le H$) meaning that the parts connecting intersection (A,B) and (C,D) is blocked. It is guaranteed that that part is a valid part of the streets or avenues, also $C - A \le 1$, and $D - B \le 1$, i.e., the part is 1 km long.

Output

For each test case, for each day, your program must output the number of ways to go to the university **modulo 2552** on a separate line. i.e., the output for each test case must contains K lines.

Sample Input	Sample Output
2 2 2 3 1 0 0 0 1 2 1 0 2 0 0 2 1 2 1 1 1 2 1 100 150 2 1 99 150 100 150 2 99 150 100 150 100 149 100 150	3 4 4 1562 0

A technical note to Java programmers

The amount of I/O for this task is quite large. Therefore, when reading input, you should avoid using java.io.Scanner which is much slower than using java.io.BufferedReader.

Problem G
Hexagonal Sticks

In this problem, we will consider an infinite hexagonal grid. The grid consists of equal regular hexagonal cells arranged in the fashion shown below. The figure also elucidates the coordinate system used to identify each cell. Each cell of the grid can be empty or blocked.

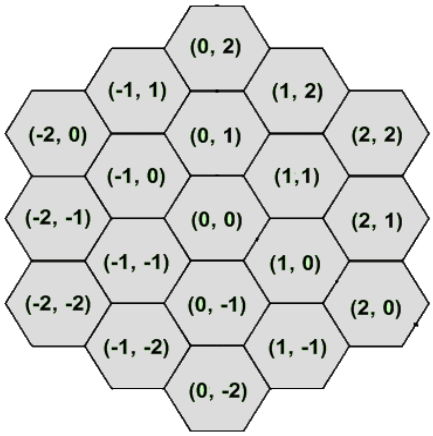
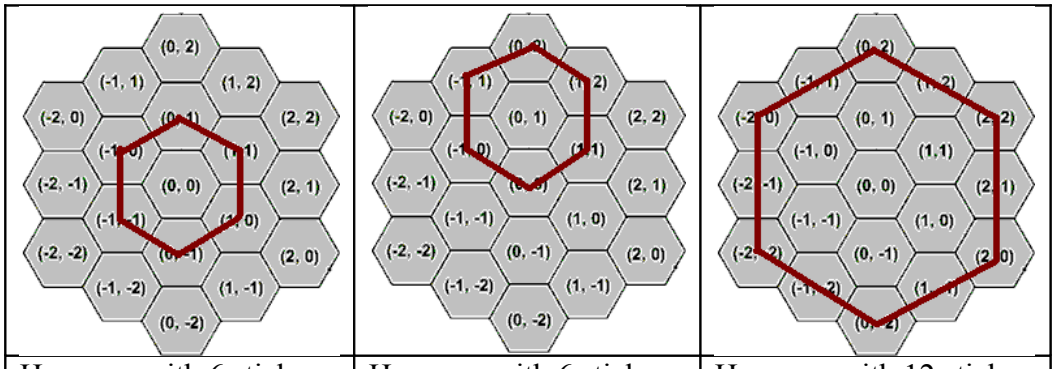


Figure: A portion of the grid

There are few sticks placed randomly on this grid. The length of each stick is one 'hexagonal unit'. This means, the end points of a stick lies on the centers of neighboring cells. Your job is to move the sticks so that a closed regular hexagonal figure is formed.
The diagrams below show some closed hexagonal figures made of sticks.

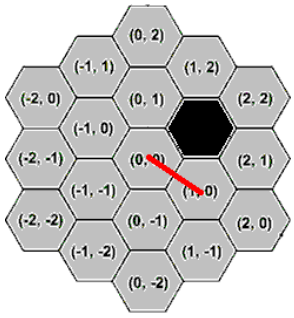


Hexagon with 6 sticks	Hexagon with 6 sticks	Hexagon with 12 sticks

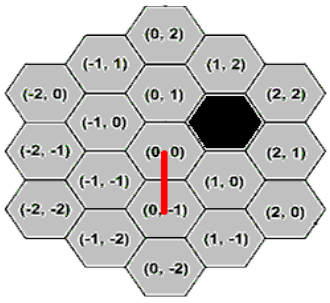
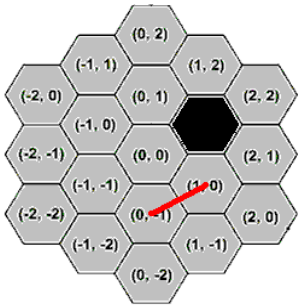
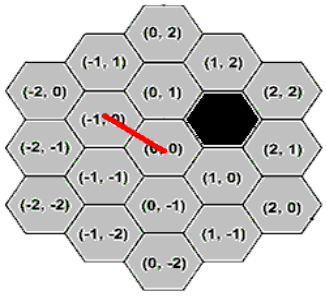
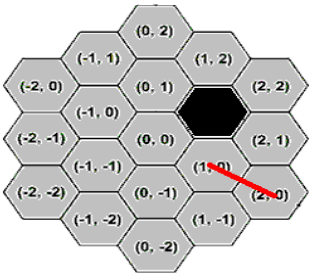
You will be given the initial coordinates of the sticks that are placed on the grid. You will also be given the coordinates of the cells that are blocked. In each move you can do one of the followings:

- Select one stick and throw it out
- Select one stick and rotate it 60 degrees clockwise/anti-clockwise about one of the end points
- Select one stick and push it along the length of the stick.

The sticks can never occupy a cell that is blocked. However, two sticks can occupy the same cells at the same time.



Consider the scenario shown above. We have an obstacle situated at coordinate (1, 1) and a stick at coordinate (0, 0) -> (1, 0). The four possible moves that can be made are depicted below

	
Rotate 60 degrees cw about (0, 0)	Rotate 60 degrees a-cw about (1, 0)
	
Push along the length	Push along the length

After all the moves are made, you have to ensure a closed regular hexagonal shape is formed with no other sticks lying around. This means the grid should contain exactly **6*x** sticks where x is positive integer. Can you do this in *minimum* number of moves?

Input

The first line of input is an integer T(T<50) that indicates the number of test cases. Each case starts with a non-negative integer S(S<9) that gives you the number of sticks available. The next S lines give you the coordinates of the sticks. The coordinates of the sticks will be of the format x1 y1 x2 y2, which means there is a stick from (x1, y1) to (x2, y2). The coordinates will be valid and the length of each stick will be one hexagonal unit as mentioned above. The next line will give you a non-negative integer B(B<20) that indicates the number of obstacles. Each of the next B lines will give you the coordinates of the obstacles. The coordinate will be of the format x1 y1. It is guaranteed that the given blocks will not overlap with any given sticks. All the coordinates (sticks and blocks) will have values in the range [-4, 4].
Note: Remember that we are dealing with infinite grids. So in the optimal result, it could be possible that the hexagonal sticks lie outside [-4, 4].

Output

For each case, output the case number first followed by the minimum number of moves required. If it is impossible to form a hexagonal grid, output "impossible" instead. Adhere to the sample input/output for exact format.

Sample Input	Sample Output
3 6 -1 -1 -1 0 -1 0 0 1 0 1 1 1 1 1 1 0 1 0 0 -1 0 -1 -1 -1 0 5 -1 0 0 1 0 1 1 1 1 1 1 0 1 0 0 -1 0 -1 -1 -1 0 7 -2 -2 -2 -1 -1 -1 -1 0 0 0 1 1 0 0 1 1 0 0 1 1 1 1 2 2 1 2 2 2 2 1 0 2 0	Case 1: 0 Case 2: impossible Case 3: 9

Problem H
Buy Your House

You are going to buy a house and hence communicated with a real estate development company, which has just started their business and you are going to be the first buyer. So they are offering you something special.

The real estate company has a rectangular shaped land of width **W** and height **H**. They are using co-ordinate system for measuring lands. (0, 0) is the lower left corner of their land and any point which has distance x from lower edge of the land and distance y from left edge of the land is known as (x, y) in the co-ordinate system.

The real estate company has already built some houses in that piece of land. All of them are rectangular shaped and their edges are parallel to edges of the main land. The location of a house can be addressed by four integers x1, y1, x2, y2. Where (x1, y1) is the lower left corner and (x2, y2) upper right corner of the house.

The special offer is that you can choose any rectangular shaped region that contains exactly one house with any amount of adjacent open space. You may not have enough money to afford open space and choose to buy only the region that a house occupies. If you have enough money, you can keep open space in front of your house for gardening!

But still there are some restrictions. For the ease of their future use of rest of their land you can only choose a rectangular shaped region and the edges of which are parallel to the edges of the main land. The corners of your selected region should be integer coordinates. It can be (3, 2) but cannot be (3.5, 2). You cannot chose a region for which part of a house is inside the region and another part of that house is outside the region. You cannot choose a region having more than one house and a region having no house. How many ways you can choose your land following the above rules?

Input

Input will start with an integer T (T is around 500), the number of test cases. Each of the test cases starts with two integers W and H (1 <= W, H <= 1000000000), width and height of the land and the next line contains an integer N (1 <= N <= 50), the number of houses in the land. Each of the next N lines will contain four integer x1, y1, x2, y2 (0 <= x1 < x2 <= W and 0 <= y1 < y2 <= H), which describes the location of the house. Note that no house can overlap with another house and all the given coordinates will be non negative integers.

Output

For each input, print a single line of the form "Case #: W", where '#' will be replaced by the case number and W will be replaced by the number of ways you can choose your land. Here W can be very large, so you should print the number of ways modulo 1000000007 as W.

Sample Input	Sample Output
2 3 3 1 1 1 2 2 10 10 2 1 1 4 4 6 6 8 8	Case 1: 16 Case 2: 429

Problem I
Synnerg Lifeform

In a science laboratory, scientists found a new kind of tiny lifeforms and named them "synnergs". Until now, few knowledge about synnergs are known. Some of these knowledge are as follow.

- There are more than one types of synnerg.
- Any new born synnerg have the same lifetime.
- Two synnergs (of certain types) can extend their lifetimes by unifying themselves together which will also transform them into a new target synnerg. The lifetime of this new target is the summation of both sources' lifetimes multiplied with an amplification factor. The type of this new target may vary from its sources. From many experiments, scientists found a set of rules to describe the unification of each synnerg pair . Some of these rules can be shown as the following table.

	Target	Source#1	Source#2	Amplification factor
1	aa	a	a	1
2	AA	a	a	17
3	bb	b	b	3
4	x	aa	bb	2
5	x	x	a	2
6	c	c	a	3

- When scientists arrange two or more of new born synnergs into a sequence, each synnerg in this sequence will try to unify itself with the one on its left or its right (and make the sequence shorter). This unification process will continue again and again (recursively). There might be lots of possible ways to unify an input sequence which also affect the lifetime of each unified synnerg.

For example of unification steps, please see the following table.

Step#	Sequence	Lifetime	Rule#	Remarks
1	a a b b	1 1 1 1	-	New born sequence
2	aa bb	2 6	1, 3	
3	x	16	4	Completely unified
1	a a b b	1 1 1 1	-	New born sequence
2	AA bb	34 6	2, 3	Final but not completely unified
1	a c a c	1 1 1 1	-	New born
2	c c	6 6	6, 6	Final but not completely unified
1	a c a c	1 1 1 1	-	New born
2	a c c	1 6 1	-, 6, -	
3	c c			Final but not completely unified

In the first example, at the first step there is a sequence of 4 new born synnergs “a a b b” where their lifetimes are “1 1 1 1” respectively. In the second step, these synnergs unify themselves into a sequence of two synnergs “aa[a a] bb[b b]” (using rule no.1 and 3) where aa's lifetime is 2=(1+1)*1 and bb's lifetime is 6=(1+1)*3. In the third step, “aa bb” unify themselves into “x[aa bb] (using rule no. 4) where its lifetime is 16=(6+2)*2.

In the second example, at the first step, the beginning sequence is the same sequence in the first example. But they alternatively unify themselves into “AA[a a] bb[b b]” (using rule no.2 and 3 instead) where their lifetimes are 34=(1+1)*17 and 6(1+1)*3 respectively. This sequence is also the final sequence which means that it is unable to unify furthermore but it is not a completely unified sequence. However, this alternative final sequence has longer life time comparing to the final sequence in the first example.

The third and fourth example also show the alternative ways of unifying the sequence “a c a c”. Please be notify that each rule is not sensitive to the order of its sources.

Goal

The scientists would like you to create a program that can find all highest lifetime synnergs that can be created by unifying a given sequence of newborn synnergs. The solution may not always be the one that is completely unified. Only part or sub-sequence unification is also acceptable.

Input

Input is a standard input which contains 2 parts of data which are separated by a blank line.

The first part is a set of unification rules.

- Each line in this first part contains one of unification rule.
- Each rule consists of 4 fields separated by spaces.
- The first 3 fields are target, source #1 and source #2 respectively. Each fields is a synnerg name which is represented by a string of at most 20 alpha-numeric characters.
- The last field is the amplification factor which is a positive integer less than or equal to 100.

The second part is a set of input sequences of synnergs.

- Each line in this second part contains one input sequence of synnergs.
- Each input sequence is a sequence of synnergs separated by spaces.

The blank line after the second part is the termination of the input.

Output

For each sequence of input, write 2 parts of output as follows

- In the first line, write the total of number of highest synnergs followed by a space and then the maximum lifetime of these synnergs.
- In the following lines, write each of the solutions in each line. Each solution contains 3 fields separated by spaces. These fields are a synnerg, its starting and finishing offset. If there are two or more solutions, they must be sorted by ascending order. The priority of comparison in sorting is the 2nd field, the 3rd field and the 1st field (or the starting offset, the finishing offset and the synnerg). For comparisons in sorting, the 2nd and 3rd field comparison is based on numerical values, and the 1st field is based on alphabetical/lexicographical (ASCII) order.

Sample Input	Sample Output
aa a a 1 AA a a 17 bb b b 3 x aa bb 2 x x a 2 c c a 3 a a b b a a b b a a a b b a a a c a c a c	1 34 AA 1 2 2 34 AA 1 2 x 1 5 1 70 x 1 6 1 6 c 1 2 1 21 c 1 3

More Explanations

There are 6 rules and 5 input sequences in this sample input.

For the first case, even though there is a completed unification “x[aa[a a] bb[b b]]” (starting from 1 to 4), its lifetime is only 16 (= {(1+1)*1 + (1+1)*3}*2) times of a newborn, which is less than 34 of “AA[a a]” starting from 1 to 2.

For the second case, there are 2 solutions. The first is “AA” starting from 1 to 2. The second is “x” starting from 1 to 5.

For the third case, there is only one solution. “x” now is the only winner (with 70 points).

For the fourth and the fifth cases, (parts of) the results has been unified according to the last rule. Since each rule is not sensitive to the order of its sources, any rule is applicable if both of its source are found. (“c c a 3” equals “c a c 3”.)

Problem J
Nowhere Money

In Nowhere town, people use “coin and slot” as their money. There are 2 types of coins called size1 and size2. Size2 coin is twice the thickness of size1. People stack their coins in slots and use them as money. There are many size of slots. The slot of size n is able to stack up n size1 coins. Only filled-up slot is considered as legal money. The value of each filled-up slot is the distinct ways its can stack coins inside. For example, for size-1 slot, there is only one way to stack a single size1 coin inside. For size-2 slot, there are 2 ways to stack two size1 coins or one size2 coin inside. And for size-5 slot, there are 8 ways to stack coins inside, which can be illustrated as follow: 1 1 1 1 1, 1 1 1 2, 1 1 2 1, 1 2 1 1, 2 1 1 1, 1 2 2, 2 1 2, 2 2 1. So the value of filled up slot with size-1, size-2 and size-5 are 1, 2 and 8 (monetary) units respectively (regardless of the type of coins or the ways they are stacked).

Mr.Thinktwice is an owner of a grocery store in this town. He noticed that customers are likely to go to the shop that can return the (money) change in the form that suits their customer. And from his little survey, he found that most customers would like to get their amount of change in the form according to these 2 simple constraints.

1. The number of slots is minimum.
2. The size of each slot must be different from each other by at least 2.

- This means that customers does not want any slots with the same size and it will be easier for them to distinguish these differences if the sizes are not too close.

So Mr.Thinktwice ask you to write a program that can give him a series of slot sizes for a given amount of change according to the previous constraints. Moreover, the series must be sorted in descending order. For more specific, any amount of change can be written in this formula.

$$X = \sum_{i=1}^{n} T(s_i) = T(s_1) + \cdots + T(s_i) + \cdots$$

where X is an amount of change.
n is the total number of slot.
s_i is the size of i^{th} slot.
T is a function mapping from a slot size to a number of distinct ways of stacking coins.
and when

For example :

$$\begin{aligned} 10 &= T(5) + T(2) \\ &= 8 + 2 \\ &= T(9) + T(2) \\ &= 34 + 1 \end{aligned}$$

Input

Input is a standard input which contains a set of integer. Each line of the input is an amount of change which represents by a positive integer less than or equal to 5,000,000,000,000,000,000 or 5×10^{18}. The input is terminated when the EOF (End-Of-File) is reached.

Output

For each amount of change, generate 4 lines of output data. The first line is the amount of change itself. The second line is a series of slot sizes (in descending order) separated by spaces. (The maximum slot size is less than or equal to 90.) The third line is a series of corresponding slot values. The fourth line is a blank line.

Sample Input	Sample Output
1 10 1000000	1 1 1 10 5 2 8 2 1000000 29 25 23 11 9 832040 121393 46368 144 55

Problem K
Highway Patrol

Crimes in city of Megacity are going high. To fight the crimes, the authorities have created a highway patrol. The city consists of a number of one-way roads. At the ends of each road, there is a base station for the patrol troops. Each base station has a number of troops. At the beginning, each station sends a troop along all the outgoing highways from that station. The troop patrols the highway and whenever it reaches the station on the other side of the highway, it waits there, and the troop that has been waiting there the most, is sent along the highway that has not been patrolled the longest time.

Soon, they faced some difficulties, cause, the frequency of patrolling a highway is more and more dependent on the number of highways that started and ended at the base station. If the number of highways started at a base station is more than the number of highways ended there, the roads are patrolled less frequently. And if, no highways end at some base station, then, the highways started from there, will not be patrolled more than once.

In this situation, the highway patrol decided to remove some highways from the patrolling schedule, so that, at each base station, the number of highways started and ended at any base station will be equal. The rest of the highways will be monitored using video surveillance. But, due to some security issues, there are some highways that have to be patrolled.

Now, given the cost of patrolling highways, and that of installing video surveillance, find the minimum cost of monitoring the whole city. Please keep in mind that, video surveillance can not substitute the highway patrol completely. So, there has to be at least one highway that will be patrolled.

Input

First line of the input contains an integer $T(T \leq 70)$, the number of test cases. This is followed by T test cases. Each test case starts with two integers $N(1 \leq N \leq 100)$ and $M(1 \leq M \leq 1000)$, the number of base stations and highways. This is followed by M lines, each containing 5 integers, $u,v,p,s,x(1 \leq u,v \leq N,0 \leq p,s \leq 1000000)$ where u and v means, the highway starts from base station u, and ends at v, p is the cost of patrolling and s is the cost of installing video surveillance. If the highway must be patrolled, then x will be one. Otherwise it will be zero.

Output

For each test case, output the case number, followed by the minimum cost to monitor the highways. If it is not possible to patrol satisfying the given constraints, output “impossible” (without quotes).

Sample Input	Sample Output
2 4 5 1 2 10 25 0 2 3 10 5 0 3 1 10 5 0 2 4 10 5 0 4 3 30 5 0 4 5 1 2 10 25 0 2 3 10 5 0 3 1 10 5 0 2 4 10 5 0 4 3 30 5 1	Case 1: 40 Case 2: 65