



ACM-ICPC Asia Phuket Regional Programming Contest 2013

Hosted by
Department of Computer Engineering
Prince of Songkla University, Phuket Campus
Phuket, Thailand

22 November 2013

Contest Problems

- There are **11** problems (A-K) to solve within 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.
- The input and output of each program are **standard input** and **standard output**.

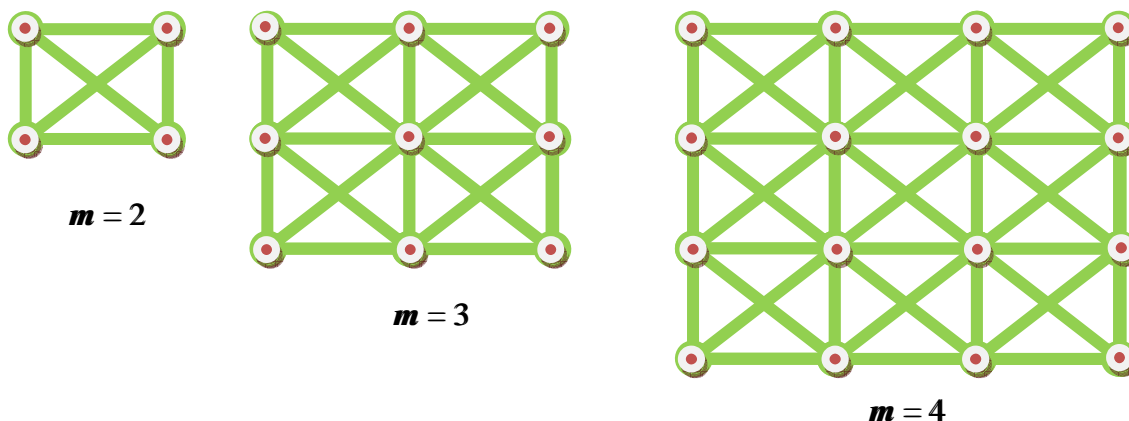
Problem A	Spanning trees in a secure lock pattern
Problem B	Teaching Hazard
Problem C	Counting Lattice Squares
Problem D	Seven Segment Display
Problem E	Airport Sort
Problem F	Boxes
Problem G	Meeting Room Arrangement
Problem H	Traveling Fool
Problem I	Cabin Baggage
Problem J	Minimal Subarray Length
Problem K	Safari Park

Problem A

Spanning Trees in a Secure Lock Pattern

Time Limit: 2s

Screen lock on a smart phone can be secured with pattern. Unlock pattern may have 9 nodes on the 3 x 3 grid. In this problem, we modify this arrangement slightly such that the grid size can be varied from 2 x 2 to $m \times m$, and only a limited number of connections between adjacent nodes are permitted. Let call this a *modified* mesh graph G and V is a set of vertices (nodes) and E is a set of edges (connections). Examples of 2 x 2, 3 x 3 and 4 x 4 modified mesh graphs are given in the figures below. Note that the corner vertices have 3 connections at most, while the middle vertices can have up to 8 connections. Only a connection to adjacent vertices is allowed, a constraint of the problem.



We define a *spanning tree* in our problem as a collection of lines in the graph forming no closed loops, containing a path between any two points of the graph covering $m^2 = n$ vertices and $n - 1$ edges. Spanning trees in a graph can have many shapes or patterns.

To calculate a number of spanning trees in G , let G be a graph with vertices labeled v_1, \dots, v_n . We form an $n \times n$ matrix tree $T = [t_{ij}]$ as follows.

- If $i = j$, then t_{ii} is the number of lines to v_i in the graph.
- If $i \neq j$, then $t_{ij} = 0$ if there is no line between v_i and v_j in G .
- If $i \neq j$, then $t_{ij} = -1$ if there is a line between v_i and v_j in G .

Then, the number of spanning trees in G is given by

$$\text{cofactor of } a_{ij} = (-1)^{i+j} M_{ij}$$

where M_{ij} is the determinant of the $(n-1) \times (n-1)$ matrix obtained by deleting row i and column j of the matrix tree T . Evaluate any cofactor of T yields the same result.

Example 1: A matrix tree T of 2 x 2 modified mesh graph is

$$T = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix}.$$

Evaluate any cofactor of T . For example, covering up row 1 and column 1, we have

$$(-1)^{1+1} M_{11} = \begin{vmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{vmatrix} = 16.$$

Therefore, a number of spanning tree is 16.

Example 2: A matrix tree T of 3 x 3 modified mesh graph is

$$T = \begin{pmatrix} 3 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 5 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & 0 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 5 & -1 & 0 & -1 & -1 & 0 \\ -1 & -1 & -1 & -1 & 8 & -1 & -1 & -1 & -1 \\ 0 & -1 & -1 & 0 & -1 & 5 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 & -1 & 0 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 & -1 & 5 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 3 \end{pmatrix}.$$

Evaluate any cofactor of T will yield the same value which is the number of spanning trees.

Hint: Determinant calculation using elementary row operation method:

To calculate a determinant you need to transform an original matrix into an upper triangular matrix (all elements below main diagonal are zero). Reduce the matrix to row echelon form using elementary row operations so that all the elements below diagonal are zero. Then multiply the main diagonal elements of a matrix to get the determinant value.

Your task is to write a program to find the number of spanning trees for a given size $m \times m$ of a modified mesh graph.

Input

The first line of the input contains an integer N ($1 \leq N \leq 5$) denoting the number of test cases. After that N test cases follow. Each test case contains a single integer m ($2 \leq m \leq 6$) denoting a size of an $m \times m$ modified mesh graph.

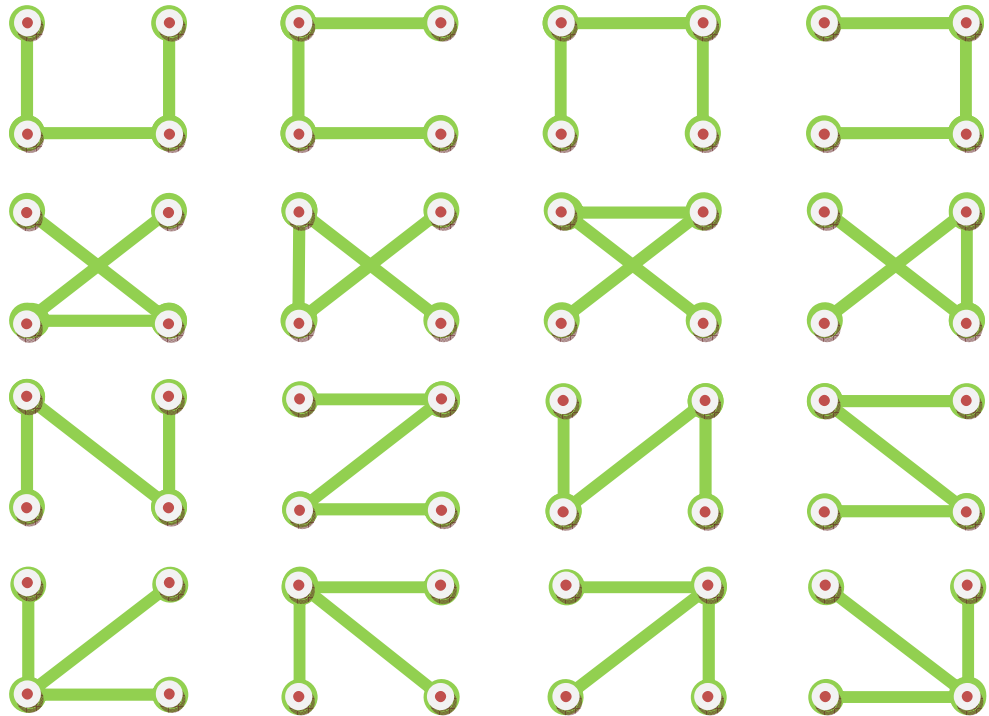
Output

For each test case, your program should output the number of spanning trees, which indicates how many patterns of spanning trees there are for a given modified mesh graph. Print out one line per test case, respectively.

Sample Input	Sample Output
1 2	16

Explanation

In the above sample input, there is only 1 test case ($N = 1$). For example, in the first test case there are 16 different spanning trees for $m = 2$. These spanning trees are listed below. Note that your program is NOT required to draw all possible patterns.



Problem B

Teaching Hazard

Time Limit: 20s

Teaching students is fun but it can often be embarrassing, which I experienced a few days ago. I was taking CSE3021 (Mathematical Analysis for Computer Science) class in my university and in the very first class I was teaching some very basic things. To be specific I was trying to teach students how to find trailing zeroes of $n!$ (factorial n) in base b . And of course many of you know that multiplicity of a prime factor p in $n!$ can be found using the formula $\left\lfloor \frac{n}{p} \right\rfloor + \left\lfloor \frac{n}{p^2} \right\rfloor + \left\lfloor \frac{n}{p^3} \right\rfloor + \dots$ to inf. This formula can also be used cleverly to find number of trailing zeroes in $n!$.

After teaching this formula I showed them how to find number of trailing zeroes in $200!$ in decimal number system and with an evil smile asked them to find out number of trailing zeroes in $100!$ in hexadecimal (16-based) number system. I knew that the correct answer is 24 and to my utter surprise I got a correct reply from a student within minutes and so I congratulated him. But a minute later when I checked his script I found that he actually calculated number of trailing zeroes in $100!$ in decimal (not Hexadecimal) number system and coincidentally that both answers (Trailing zeroes in hexadecimal and decimal number system) were 24. So I was a bit embarrassed and now you have to help me find out why those two answers were same? Given a number n , you will have to find how many pair of bases (b_1, b_2) are there for which $n!$ (Factorial n) has exactly p trailing zeroes in both base b_1 and base b_2 . Here p is a positive integer not less than x .

Input

Input file contains 1000 lines of inputs. Each line contains two integers n ($1 \leq n \leq 100000$) and x ($2 \leq x \leq 2500$). Input is terminated by a line containing two zeroes.

Output

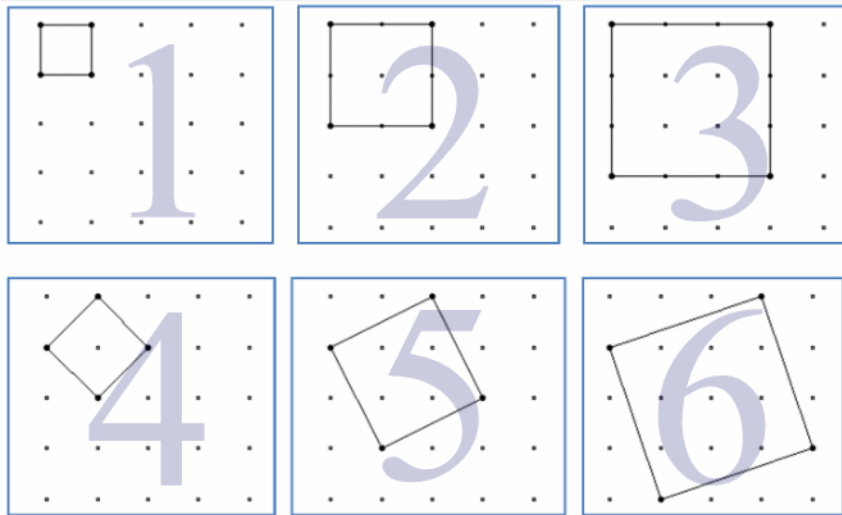
For each line of input produce one line of output. This line contains an integer which denotes number of base pairs (b_1, b_2) so that $n!$ has exactly p trailing zeroes in both bases where p is not less than x . You can assume that inputs will be such that none of the output numbers will exceed $5 \cdot 10^{18}$.

Sample Input	Sample Output
6 2	6
9 3	2
0 0	

Problem C

Counting Lattice Squares

Time Limit: 5s



A lattice square is a square whose vertices are all lattice points. A lattice-point is a point in Cartesian coordinate system whose abscissa and ordinate are integers. For example (1, 5) is a lattice point but (1, 1.5) is not. In a (n x m) grid there can be many lattice squares. In the figure on the left you can see some lattice squares in a (4x4) grid. Counting lattice squares with axis-parallel sides (Figure 1, Figure 2 and Figure 3) is trivial but there are lattice

squares whose sides are not axis parallel (Figure 4, Figure 5 and Figure 6) and counting them is just a bit harder. Some of these squares can have even area (Figure 2, Figure 4, Figure 6) and some others can have odd area (Figure 1, Figure 3, Figure 5). Given an (m x n) grid your job is to write a program that counts how many different lattice squares with odd area can be drawn in that grid. Two lattice squares are different if they do not share all four vertices.

Input

The input file contains at most 50000 lines of inputs. Each line contains two integers m, n ($1 \leq m, n \leq 100000$). Input is terminated by a line containing two zeroes.

Output

For each line of input produce one line of output. This line contains an integer S, which denotes how many different lattice squares with odd area can be drawn in an (m x n) grid. You can assume that the value of S fits in a 64-bit signed integer.

Sample Input	Sample Output
2 2	4
3 3	12
4 4	28
0 0	

Problem D

Seven Segment Display

Time Limit: 1s

Seven segment numeric displays are ubiquitous. It uses seven segments to display numbers.

Here is a figure which depicts all the segments used in a typical seven segment display (We'll be using the acronym SSD for convenience from now on).

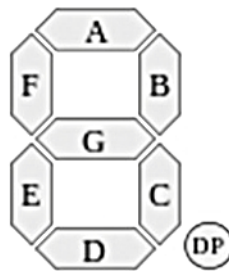


Figure 1: Segments used for SSD representation.

Here, DP represents decimal place which is not necessary in the context of this problem.

And here are the numbers from 0 to 9 represented in SSD.



0 uses segments A, B, C, D, E, F

1: B, C

2: A, B, G, E, D

3: A, B, C, D, G

4: B, C, F, G

5: A, C, D, F, G

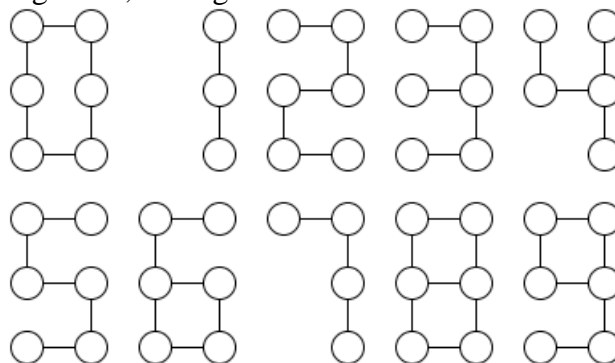
6: A, C, D, E, F, G

7: A, B, C

8: A, B, C, D, E, F, G

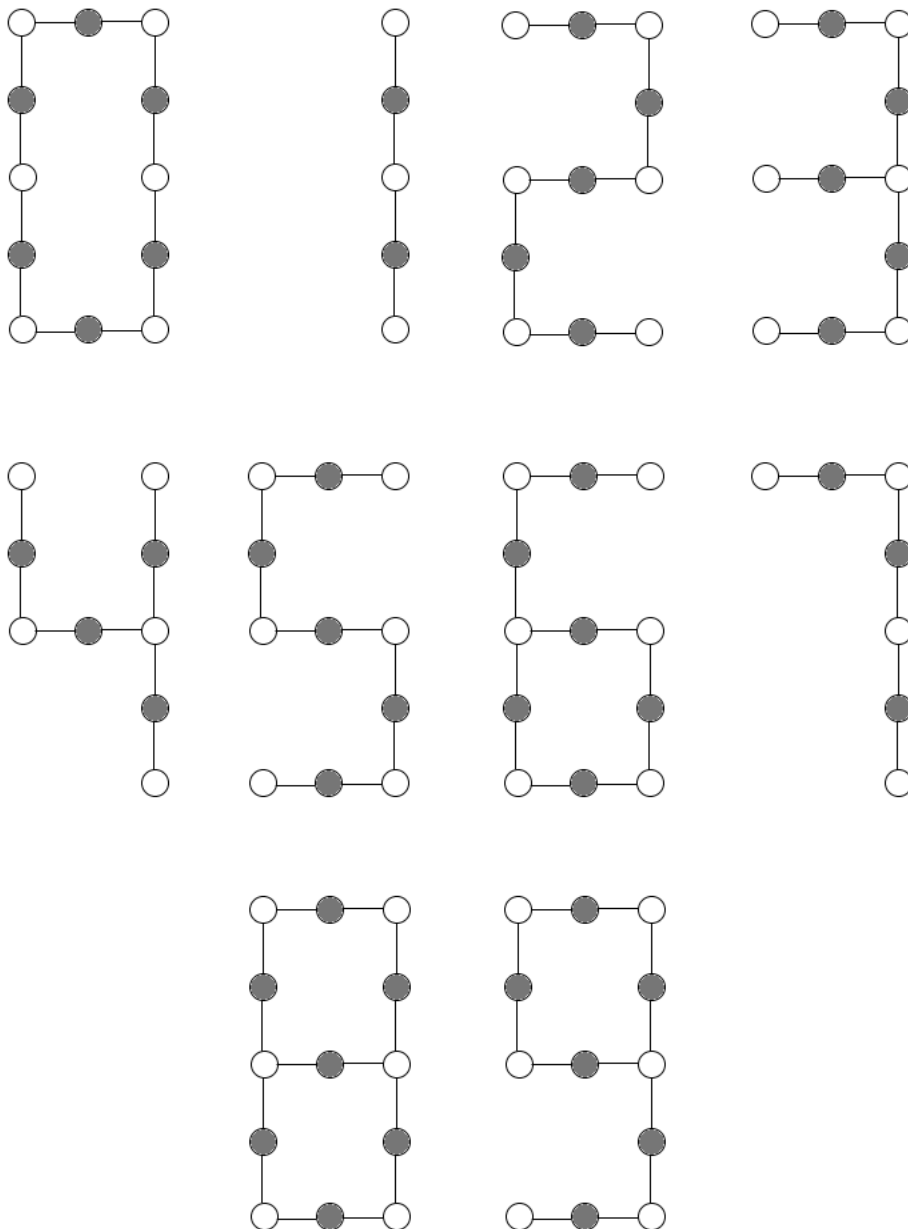
9: A, B, C, D, F, G

Now, imagine the SSD representation of a digit as a graph. The endpoints of the segments are the nodes and segments are edges. So, the digits will look like:



We call this representation a 0-degree SSD graph. A k -degree ($k > 0$) SSD graph is made by dividing each edge of a 0-degree graph into $k+1$ edges and introducing k nodes in between them.

To explain more, 1-degree graphs of all digits are shown below. The darker nodes are the newly introduced nodes.



You'll be given a graph with n nodes and m edges. You'll need to print all the (degree, digit) pairs for which the given graph is valid.

Input

The first line of the input contains an integer which denotes the number of test cases T ($1 \leq T \leq 20$). T sets of case will follow. Each case will start with a couple of numbers n ($1 \leq n \leq 500$) and m ($1 \leq m \leq 1000$)- the number of nodes and the number of edges respectively. Each of the next m lines will contain a pair of numbers (u, v) meaning that there is an edge from node u to node v . Nodes are numbered from 1 to n . It's guaranteed that there is no duplicate or self-edges in the input.

Output

For each set of inputs, output one set of output. First line of a set should be of the format, Case X: Y (here, X is the serial of the input and Y is the number of (digit, degree) pairs) in a line. Then print each (digit, degree) pair - one pair in each line. The pairs should be sorted according to digit first then degree. Each number in a pair should be separated with a space. Print a blank line between consecutive test cases.

Sample Input

```
2
16 15
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 10
10 11
11 12
12 13
13 14
14 15
15 16
4 3
1 2
1 3
1 4
```

Sample Output

```
Case 1: 3
2 2
5 2
7 4
```

```
Case 2: 0
```

Problem E

Airport Sort

Time Limit: 2s

There are few airlines that don't specify seat numbers before boarding. Instead, each passenger gets a ticket containing a unique integer in the range $[1, n]$ where n indicates the total number of seats in the airplane. Each ticket number belongs to a specified zone. Suppose a zone contains k tickets, then tickets with numbers $[1, k]$ are in zone 1; tickets with numbers $[k+1, 2k]$ are in zone 2 and so on. The last zone may contain less than k tickets if n is not divisible by k .



Before boarding, all the n passengers line up randomly in a straight line. In order to expedite the boarding process, it is convenient that the first k passengers in the line belong to zone 1, the next k in zone 2 and so on. There are two ways of rearranging the positions of the passengers.

- 1) Adjacent passengers keep swapping places until the required order is accomplished. Every second, only one pair of adjacent passengers can swap their places.
- 2) All the passengers walk simultaneously towards their correct positions. To walk from position x to position y , it takes $|x-y|$ seconds. $|x-y|$ is the absolute difference of x and y .

As you can guess, the first approach takes more time, but it's less 'noisy'. For this problem, you have to determine how much faster the 2nd approach is. More specifically, suppose the minimum time required to rearrange the passengers using the first approach is X and the minimum time for the second approach is Y , you have to find $X - Y$.

As an example, consider a plane of capacity 10 ($n = 10$) and zones of size 3 ($k = 3$). Suppose the passengers line up in the following order

3 7 1 2 4 6 5 8 10 9

Each integer represents the ticket number of the corresponding passenger. So, for the above example, the first passenger in the line has a ticket with number 3. Passengers with ticket numbers 7, 2, 5, 10 and 9 are not in their right positions.

Using the first strategy, it'd take a minimum of 6 swaps (and hence 6 seconds) to achieve the desired order.

- I. swap 7 with 1 \Rightarrow 3 1 7 2 4 6 5 8 10 9
- II. swap 7 with 2 \Rightarrow 3 1 2 7 4 6 5 8 10 9

III. swap 7 with 4 => 3 1 2 4 7 6 5 8 10 9

IV. swap 7 with 6 => 3 1 2 4 6 7 5 8 10 9

V. swap 7 with 5 => 3 1 2 4 6 5 7 8 10 9

VI. swap 10 with 9 => 3 1 2 4 6 5 7 8 9 10 (now everyone is where they are supposed to be)

Using the second strategy, it'd take a minimum time of 5 seconds if we decide the final order to be (3 2 1 4 6 5 7 8 9 10). Here passengers with ticket 3, 1 and 8 are already in their right positions; it takes 1 second for passengers with tickets 4, 5, 6, 9 and 10 to get to their right positions; it takes 2 seconds for passenger with ticket 2 to come to her final position; it takes 5 seconds for passenger with ticket #7 to get to her right position. So this means, after 5 seconds, everyone will be at their right places using the 2nd strategy. There are other arrangements for which we get 5 seconds, but it's not possible to come up with something less.

So 2nd strategy is 1 second faster.

Input

The first line of input is an integer **T** ($T < 50$) that indicates the number of test cases. Each case consists of 2 lines. The first line contains 2 positive integers **n** ($n \leq 20000$) and **k** ($k \leq n$). The meanings of these variables are mentioned above. The next line contains **n** distinct integers in the range [1, **n**]. The first integer represents the ticket number of the passenger standing in front of the queue.

Output

For each case, first output the case number followed by the required result (i.e. how much faster the 2nd approach is).

Sample Input	Sample Output
3	Case 1: 1
10 3	Case 2: 0
3 7 1 2 4 6 5 8 10 9	Case 3: 4
11 3	
1 2 3 4 5 6 7 8 9 10 11	
5 2	
5 4 3 2 1	

Problem F

Boxes

Time Limit: 5s

Given n boxes with widths of w_1, w_2, \dots, w_n and another big box with width W , find how many ways the boxes can be put in the big box. The constraints are:

- 1) Of course the summation of widths of the placed boxes should not be greater than W .
- 2) The boxes should be placed one by one starting from left without leaving any empty spaces between them. So, the end of the big box may contain empty spaces. But if there is any unplaced box which can be fit in this space, the ordering should be considered invalid (See the explanation for sample case 1).
- 3) Two orderings are considered different if in one ordering, one box is in i^{th} position, but in another ordering, it isn't.
- 4) If two boxes have same widths, they should be considered same.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts two integers n ($1 \leq n \leq 100$) and W ($1 \leq W \leq 1000$). The next line contains n space separated integers, denoting $w_1 w_2 \dots w_n$ ($1 \leq w_i \leq W$).

Output

For each case, print the case number first and the result modulo **10007**.

Sample Input	Output for Sample Input
2	Case 1: 6
3 5	Case 2: 30
1 2 3	
5 10	
1 2 2 4 5	

Notes

For the first case, the orderings are

1 2
1 3
2 1
2 3
3 1
3 2

Only 1 or 2 or 3 is not solutions since we can still place another box.

Problem G

Meeting Room Arrangement

Time Limit: 1s

Faculty of Engineering of PSU has a large meeting room for faculty staff to organize events and meetings. The use of the meeting room must be reserved in advance. Since the meeting room is available in 10 hours per day and there may be several events that want to use the meeting room, the best usage policy is to maximize the number of events in day.

Suppose that the meeting room is available from time 0 to 10 (10 hours). Given the list of start time and finish time of each candidate event, you are to write a program to select the events that fit in the meeting room (i.e. their times do not overlap) and give the *maximum* number of events in a day.

Input

The first line is a positive integer n ($1 \leq n \leq 100$) which determines the number of days (test cases). Each test case consists of the time of the candidate events (less than 20 events). Each event time includes 2 integers which are start time(s) and finish time(f), $0 \leq s \leq 9$, $1 \leq f \leq 10$ and $s < f$. The line containing 0 0 indicates the end of each test case. Note that an event must use at least 1 hour.

Output

For each test case, print out the maximum number of events that can be arranged in the meeting room.

Sample Input	Sample Output
3	4
0 6	4
5 7	1
8 9	
5 9	
1 2	
3 4	
0 5	
0 0	
6 10	
5 6	
0 3	
0 5	
3 5	
4 5	
0 0	
1 5	
3 9	
0 0	

Problem H

Traveling Fool

Time Limit: 10s

You must have heard the ‘Traveling Salesman Problem’. Here we are talking about another problem named ‘Traveling Fool Problem’. Assume that there are n cities connected by m one way roads. Each road is labeled by an uppercase English letter (i.e ‘A’ to ‘Z’). There can be multiple roads between two cities but no roads will start and end at the same city.

The traveling fool starts his journey from city s and he continues his journey until he reaches t , or he reaches a city from which t is unreachable. If he is in city u , he can choose any road that starts from u with equal probability.

He may visit same city/road more than once, but once he reaches t , he immediately stops his journey and remembers the road-labels he found in his path in the same order the roads were visited. If the road-labels in the path he traveled form a palindrome, he finds himself lucky. If he is unable to reach t or the road-labels don’t form a valid palindrome, he finds himself unlucky.

Given the cities, roads, s and t , can you find the probability of Mr Traveling Fool being lucky?

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers n ($2 \leq n \leq 12$) and m ($0 \leq m \leq 1000$). Each of the next m lines contains two integers, u v ($0 \leq u, v < n, u \neq v$) and an uppercase English letter w , meaning that there is a one-way road from city u to city v and the road label is w . Next line contains an integer ($1 \leq q \leq 150$) denoting the number of queries. Each of the next q lines contains two integers denoting s t ($0 \leq s, t < n, s \neq t$).

Output

For each case, print the case number first. Then for each query, print the probability as stated. Errors less than 10^{-4} will be ignored.

Sample Input	Sample Output
2	Case 1:
4 3	1.000000
0 1 A	0.000000
1 2 A	Case 2:
2 3 A	0.000000
2	0.333333
0 3	
2 0	
5 4	
1 2 B	
2 3 D	
2 4 A	
2 0 B	
2	
1 3	
1 0	

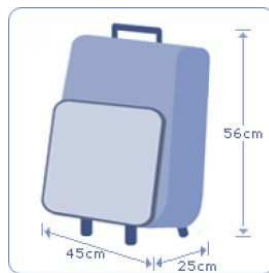
Problem I. Cabin Baggage

Time Limit: 1s

Cabin baggage (also called carry on or hand baggage) is a bag that a passenger is allowed to bring into an aircraft. For safety purpose, cabin baggage must not be too heavy or too big. Every airline company can set its own size and weight limits of cabin baggage according to the IATA (International Air Transport Association) guidelines.

ICPC airline has set the cabin baggage limits as follows:

Cabin baggage should have a maximum length of 56 cm, width of 45 cm and depth of 25 cm or the sum of all 3 sides (length+width+depth) should not exceed 125 cm. Its weight must not exceed 7 kg.



The company has a laser measurement device with high precision to evaluate the size and weight of cabin baggage. The device gives 4 values which are positive numbers with 2 decimal points. They are length, width, depth (in cm) and weight (in kg), respectively.

For example, 51.23 40.12 21.20 3.45 (this bag is allowed)
 60.00 30.00 20.00 7.00 (this bag is allowed)
 52.03 41.25 23.50 7.01 (this bag is not allowed)

Your task is to write a program to check whether or not a cabin baggage is allowed.

Input

The first line contains an integer t ($1 \leq t \leq 300$) which determines the number of test cases (i.e. cabin baggage to verify). The following t lines contain the measurement of cabin baggage. Each line contains 4 values which are length, width, depth and weight, respectively. All these 4 values are positive numbers with 2 decimal points.

Output

For each test case, print out in a line, 1 if it is allowed or 0 otherwise. Finally, print out the total number of the allowed bags in the last line.

Sample Input	Sample Output
4	1
51.23 40.12 21.20 3.45	1
60.00 30.00 20.00 7.00	0
52.03 41.25 23.50 7.01	0
50.00 45.00 30.10 6.02	2

Problem J

Minimal Subarray Length

Time Limit: 3s

You are given an integer sequence of length N and another value X . You have to find a contiguous subsequence of the given sequence such that the sum is greater or equal to X . And you have to find that segment with minimal length.

Input

First line of the input file contains T the number of test cases. Each test case starts with a line containing 2 integers $N(1 \leq N \leq 500000)$ and $X(-10^9 \leq X \leq 10^9)$. Next line contains N integers denoting the elements of the sequence. These integers will be between -10^9 to 10^9 inclusive.

Output

For each test case output the minimum length of the sub array whose sum is greater or equal to X . If there is no such array, output -1.

Sample Input	Sample output
3 5 4 1 2 1 2 1 6 -2 -5 -6 -7 -8 -9 -10 5 3 -1 1 1 1 -1	3 -1 3

Problem K

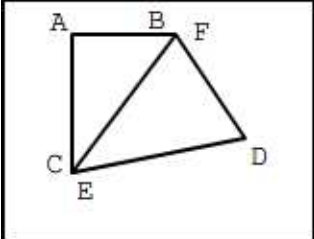
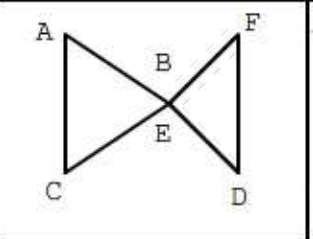
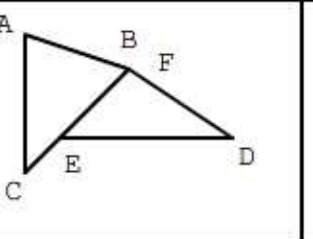
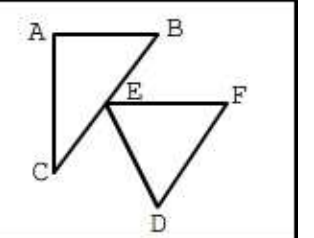
Safari Park

Time Limit: 5s

Have you ever been to safari park? It is a zoo-like place where animals are not kept under cage. Visitors may drive or walk through safari and observe free animals. Even sometimes dangerous animal like lion, tiger are also kept free. I have also opened a safari park of my own. Animals live in different regions here. Region of one animal does not intersect with regions of other animal for safety purpose. But two regions may have same boundary though it is not allowed to have the corner point on any other boundary. The animals are trained so that they do not leave their own territory. Now the safari park is so big that a single map cannot show the whole place. So I decided to make a device to help people to find the places of their interest. However, to reduce the cost for device I bought some low end one. Their memory is quite low. To make things less complicated and efficient from processing perspective, we have made all the regions triangle. But that is not all. All the regions will not be loaded initially. Regions will be loaded depending on the visitor's place. Visitor may visit from one place to another place park by mono-rail. So the triangles may appear in the device in different places. Since the device is not that much advance, it can show the regions but it does not show any label. One has to enter the co-ordinate of a point in a text box and the device will show the label of that region in an alert message. There are two special labels. 0 denotes outside of any region, -1 denotes on boundary / corner.

For example, suppose currently we know about two regions: Giraffe: (0, 0), (0, 5), (5, 0) and Dolphin: (10, 10), (10, 5), (5, 10). Now if the query co-ordinate is: (1, 1) then it will show Giraffe. If the query is for (9, 9) the answer will be Dolphin. However if the query is: (5, 5) it will show 0. Now suppose a new region appears in the map: Lion: (4, 6), (6, 4), (4, 4). Now again if the query is (5, 5) the answer will be -1.

So to summarize, you will be given some triangles. Triangle may share side and also corners, but no triangle will have its corner on any sides of any other triangle, i.e. sides of different triangles can be identical, but not partially overlapping. You may check the following diagram for clarity:

			
Sides completely overlap. Valid.	Sharing Corners. Valid.	Side partially overlap. Invalid.	Corner on border. Invalid.

No two triangles will have common region. If a query point is strictly inside some region you should print its label, if it is strictly outside all regions you should print 0 and if it is on some boundary / corner you should print -1. Details of input and output can be found in corresponding section.

Input

First line of the test file contains a positive integer T (≤ 2) which is the number of test cases. First line of each case contains N ($\leq 300,000$) denotes number of commands. Each command will start with a **Q** or **R**. **Q** stands for query and **R** stands for region. **Q** will be followed by 2 integers: x_q and y_q . **R** will be followed by 6 integers: x_1 y_1 x_2 y_2 x_3 y_3 . You may assume that there won't be more than 50,000 **R** commands.

However these numbers will not give you the real co-ordinate. You have to decode the given co-ordinates using the answer to last query. Suppose d is the answer to the last query **Q** (Initially $d = 0$), then real co-ordinate of given co-ordinate x, y will be: $x + x_1[d]$, $y + y_1[d]$. Here, $x_1[d]$, $y_1[d]$ is the first co-ordinate (decoded) of d 'th region. d 'th region is the region provided by d 'th **R** command. However, if $d = 0$ or -1 (in case of out of region or on boundary / corner) you should consider

$x_1[d] = y_1[d] = 0$. A query should be answered considering only the regions provided before the corresponding **Q** command. If there are **n** **R** commands before a **Q** command then answer to this command will range from **-1** to **n**, **-1** if on boundary / corner, **0** if outside of any region and other number depending on the region in which the point is in. You may assume that the co-ordinates of the regions will always be in clock wise order. You may also assume that the regions will appear in completely random order even though the regions may not be random themselves. Value of decoded **x y** co-ordinates are non negative and will be bounded by **100,000**.

Output

For each test case first print case number. Hence for each **Q** command print the answer of the query. Do not print any blank line between test cases. For details follow the sample input output.

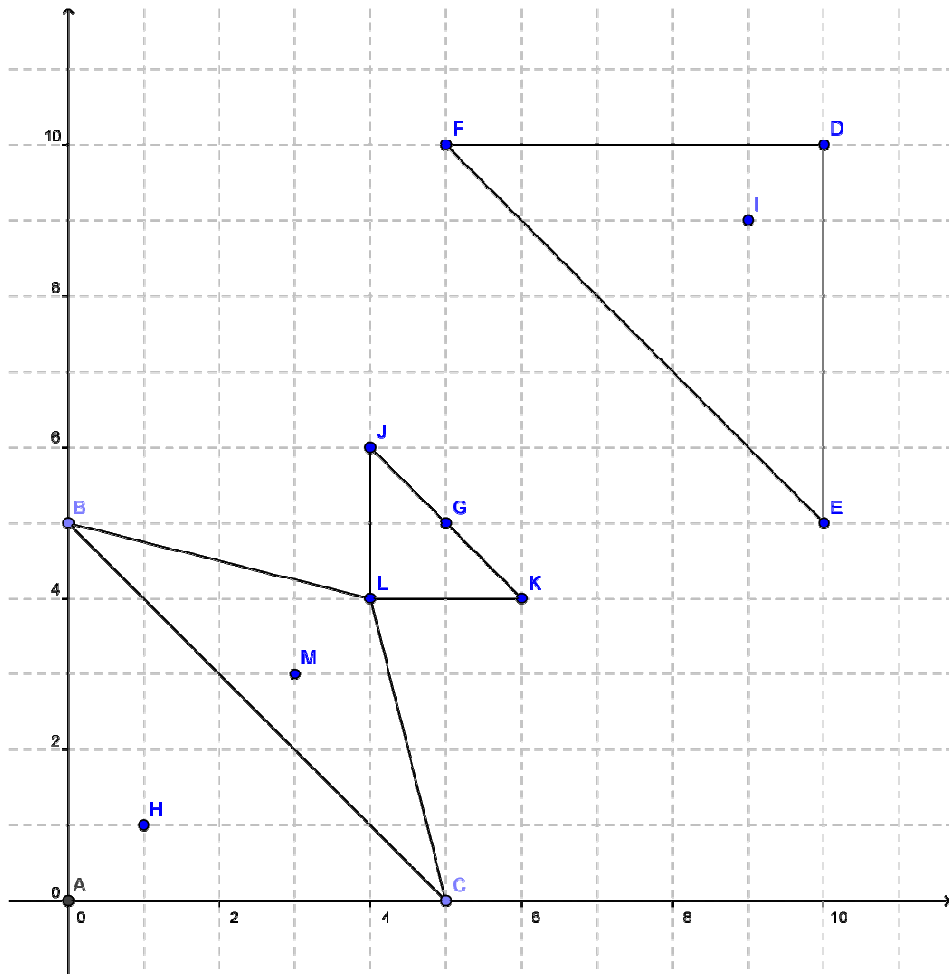
Sample input

```
1
10
R 0 0 0 5 5 0
R 10 10 10 5 5 10
Q 5 5
Q 1 1
Q 9 9
R -6 -4 -4 -6 -6 -6
Q -5 -5
Q 1 1
R 0 5 4 4 5 0
Q 3 3
```

Sample Output

```
Case 1:
0
1
2
-1
1
4
```

Explanation of sample input:



Sample input	Explanation
8	Initially $d = 0$
R 0 0 0 5 5 0	Since $d = 0$ and $x_1[d] = 0$, $y_1[d] = 0$ Decoded co-ordinates: 0 0 0 5 5 0 In the picture it is ABC triangle
R 10 10 10 5 5 10	Since $d = 0$ and $x_1[d] = 0$, $y_1[d] = 0$ Decoded co-ordinates: 10 10 10 5 5 10 In the picture it is DEF triangle
Q 5 5	Since $d = 0$ and $x_1[d] = 0$, $y_1[d] = 0$ Decoded co-ordinates: 5 5 This is out of region. Answer is 0 and so $d = 0$. In the picture it is G point
Q 1 1	Since $d = 0$ and $x_1[d] = 0$, $y_1[d] = 0$ Decoded co-ordinates: 1 1 This is in 1st region. Answer is 1 and so $d = 1$. In the picture it is H point
Q 9 9	Since $d = 1$ and $x_1[d] = 0$, $y_1[d] = 0$ Decoded co-ordinates: 9 9 This is in 2nd region. Answer is 2 and so $d = 2$. In the picture it is I point

R -6 -4 -4 -6 -6 - 6	Since $d = 2$ and $x_1[d] = 10$, $y_1[d] = 10$ Decoded co-ordinates: 4 6 6 4 4 4 In the picture it is JKL triangle
Q -5 -5	Since $d = 2$ and $x_1[d] = 10$, $y_1[d] = 10$ Decoded co-ordinates: 5 5 This is on the boundary of 3rd region. Answer is -1 and so $d = -1$. In the picture it is G point again
Q 1 1	Since $d = -1$ and $x_1[d] = 0$, $y_1[d] = 0$ Decoded co-ordinates: 1 1 This is in 1 st region. Answer is 1 and so $d = 1$. In the picture it is H point again
R 0 5 4 4 5 0	Since $d = 1$ and $x_1[d] = 0$, $y_1[d] = 0$ Decoded co-ordinates: 0 5 4 4 5 0 In the picture it is BLC triangle
Q 3 3	Since $d = 1$ and $x_1[d] = 0$, $y_1[d] = 0$ Decoded co-ordinates: 3 3 This is in 4 th region. Answer is 4 and so $d = 4$. In the picture it is M point

The following picture should give you an idea of how one of the judge test case look like:

