The 2008 ACM ASIA Programming Contest Dhaka Site

Sponsored by IBM

Hosted by North South University
Dhaka, Bangladesh

8th November, 2008
You get 19 Pages
10 Problems
&
300 Minutes

Rules for ACM-ICPC 2008 Asia Regional Dhaka Site:

a) Solutions to problems submitted for judging are called runs. Each run is judged as accepted or rejected by a judge, and the team is notified of the results.

b) Notification of accepted runs will **NOT** be suspended at the last one hour of the contest time to keep the final results secret. Notification of rejected runs will also continue until the end of the contest. But the teams will not be given any balloon and the public rank list will not be updated in the last one hour.

c) A contestant may submit a clarification request to judges. If the judges agree that an ambiguity or error exists, a clarification will be issued to all contestants.

d) Contestants are not to converse with anyone except members of their team and personnel designated by the organizing committee. Systems support staff may advise contestants on system-related problems such as explaining system error messages.

e) While the contest is scheduled for a particular time length (five hours), the Chief Judge has the authority to alter the length of the contest in the event of unforeseen difficulties. Should the contest duration be altered, every attempt will be made to notify contestants in a timely and uniform manner.

f) A team may be disqualified by the Chief Judge for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, or distracting behavior.

g) Ten problems will be posed. So far as possible, problems will avoid dependence on detailed knowledge of a particular applications area or particular contest language. Of these ten problems at least two will solvable by first year computer science student, two will be solvable by a second year computer science student and rest will determine the winner.

h) Contestants will have foods available in their contest room during the contest. So they cannot leave the contest room during the contest without permission from the room invigilator. From a single room, members of only one team can get out at a time for maximum 15 minutes. The contestants are not allowed to communicate with members of any other teams or coaches while they are outside the room. In other words rule (d) and (f) is valid within or outside the contest room.

i) Team can bring upto 200 pages of printed materials with them but they can also bring three additional books. But they are not allowed to bring calculators or any machine readable devices like CD, DVD, Pen-drive, IPOD, MP3/MP4 players, floppy disks etc.

j) With the help of the volunteers the contestants can have printouts of their codes for debugging purposes.

k) The decision of the judges is final.

l) Teams should inform the volunteers if they don't get reply from the judges within 15 minutes of submission. Volunteers will inform the floor supervisor. Teams should also notify the volunteers if they cannot log in into the PC^2 system. This sort of complains will not be entertained later.



A Find the Format String

Input: Standard Input
Output: Standard Output

Reading input is a very important and easy but boring part of programming. So functions which can take difficult sort of inputs very easily are very popular among programmers. Such a function is the ancient scanf() function of C, which exists even today with all its popularity and glory.

The main strength of scanf() function is its format strings. For example "%d" is used to read

output the ASCII-graphically smallest one. (Lexicographically means word ordering as if they were in lexicon or dictionary. So “and” is smaller than “bbt” because “and” would have appeared before “bbt” if they both were kept in a dictionary. ASCII-cographically sorted means sorted in the order in a lexicon where all the ascii characters are the alphabets. So here “abc” is smaller than “cde”, “Mccain” is smaller than “barak” etc. The format string must contain at least one alpha-numeral character and should contain no alpha-numeral more than once. If no format string of positive length and consisting only alpha-numerals can produce such output or the input output pair seems impossible then print the line “I_AM_UNDONE” instead (without the quotes). Note that the format string is printed enclosed in a third bracket. Look at the output for sample input for formatting details.

Sample Input	Output for Sample Input
5 “11” “11” “243” “24” “563” “56” “784” “784” “789” “78” 1 “A” “b” 0	Case 1: [01245678] Case 2: I_AM_UNDONE

B Switch Bulbs

Input: Standard Input
Output: Standard Output

You are given n bulbs and m switches. Each of the switches toggles a list of bulbs. Initially all the bulbs are turned off. Now for a set of desired states of the bulbs calculate the minimum number of switch presses required to reach that state.

Input

Input contains multiple test cases. First line contains an integer T the number of test cases. Each test case starts with a line containing 2 integers n ($1 \le n \le 15$) and m ($1 \le m \le 40$). Next m line contains the description of m switches. Each of these lines starts with an integer k denoting the number of bulbs that toggles their states after pressing this switch. The rest of the line contains k distinct integers denoting the indices of the bulbs. The bulbs are numbered from 0 to n-1. The next line contains an integer q($1 \le q \le 5000$) that denotes the number of queries. Each of the following q line contains a binary string of length n denoting the desired states of the n bulbs: 1 means the bulb must be on and 0 means the bulb must be off. The rightmost character is the state of bulb 0 and the leftmost character is the state of bulb n-1.

Output

For each test case output contains q+2 lines. First line contains "Case x:" where x is the number of test cases. Each of the next q lines contains one integer denoting the minimum number of switch presses required to reach the bulb states in the i'th query. If the state cannot be reachable by a series of switch presses output -1. The last line will be a blank line.

Sample Input	Output for Sample Input
2 3 3 3 0 1 2 2 1 2 1 2 3 101 010 111 4 5 1 0 1 1 2 2 3 3 0 1 3 2 2 3 3 1111 1010 0101	Case 1: 3 2 1 Case 2: 3 2 3

C Schedule of a Married Man

Input: Standard Input
Output: Standard Output

Last year we set a problem on bachelor arithmetic which made some bachelors really unhappy. So to even things up, we are making a problem on the tough schedule of a married man.

Our dashing hero Danny has recently got married and that has created a lot of problems for him, at least that is what his friends think. So many broken promises, so many missed appointments and dinners. Err! Danny, now is losing tracks of even simplest of calculations, so you must help him to decide whether he can attend his meeting or not. Danny is busy with his wife for a large portion of the day. This large portion is denoted by a starting time and an ending time. Then Danny has an important meeting in a day, he misses that if it overlaps or touches (For example, if Danny's time span with his wife finishes at 18:00 and the meeting starts at 18:00 then the two schedules conflict and Danny misses the meeting) the time scheduled for his wife. Given the time span Danny has allotted for his wife and the time span of the meeting you, will have to find whether Danny misses that meeting or not.

Input

First line of the input file contains an integer N (0<N<2001) which denotes how many sets of inputs are there. The input for each set is given in two lines. The description for each set is given below:

First line of each set contains two strings separated by a single space. These two strings denote the time span Danny is busy with his wife. The second line also contains two strings which denotes the time when Danny has to attend a meeting. All the strings that denote time are of the format hh:mm (two digit for hour and two digit for minute). For example "forty five past eight" (Morning) is denoted as 08:45, "forty five past 9" (night) is denoted as 21:45. You can assume that all times are valid 24-hour clock time, starting time strictly precedes ending time and all times are within a single day.

Output

For each set of input produce one line of output. This line contains the serial of output followed by a string which denotes Danny's decision. If Danny can attend the meeting then print "Hits Meeting" and if Danny misses (Mrs) the meeting as it conflicts with the time allotted for his wife print "Mrs Meeting" instead.

Sample Input

```
3
17:47 22:40
06:18 17:04
10:44 17:05
01:11 01:27
03:36 19:02
14:33 15:24
```

Output for Sample Input

```
Case 1: Hits Meeting
Case 2: Hits Meeting
Case 3: Mrs Meeting
```

D Puzzles of Triangles

Input: Standard Input
Output: Standard Output

Puzzles with geometric shapes are very interesting and is said to have great impact in developing geometric insight of a child. ACM (Actual Challenge Makers) is planning to bring out one such puzzle. The specialty of this puzzle is that all the pieces of this puzzle have the shape of right-angled triangles. All these right angled pieces make the target rectangular shape and it is shown in figure 2.

It may be a bit difficult to construct or even reconstruct a puzzle which is only composed of pieces with the shape of a right-angled triangle. However if the actual rectangular puzzle is divided into several square sub-regions, and then those sub-regions are divided into four right-angled pieces then such puzzles are easy to make and solve.

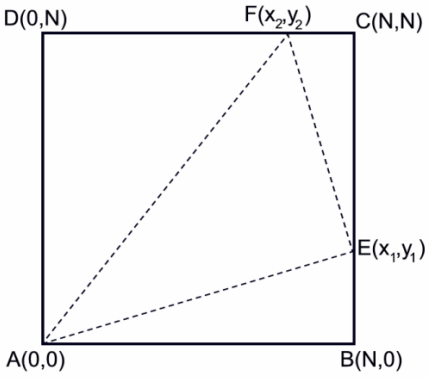


Figure 1: A cut to make four right angled triangle shaped pieces.

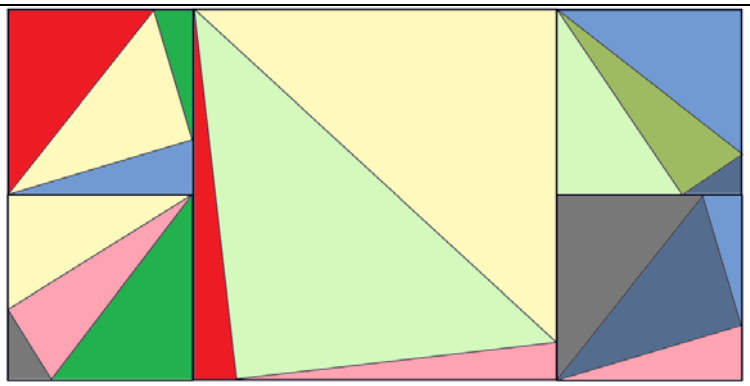


Figure 2: A conventional puzzle of triangles.

Whether a rectangle can be divided into small pieces of squares can itself be an interesting task to solve. But that is not the problem you have to solve here. Your job is to decide whether a square shaped sheet can be divided into four right-angled triangles as shown in figure 1. The machine that is used to cut the pieces can only deal with integers. So the length of the square shaped sheets is always expressed in integer units and also cutter can only cut in a straight line from one integer coordinate to another. The cutting must start on one of the corners of the square shaped sheet, continue cutting in a triangular path (Like AEF showed in Figure 1) and again finish at the corner it started. And if four corners of the square sheet are lattice points (0,0), (N,0), (N,N) and (0,N), where N is the length of sides of the square shaped sheet then point E and F should also be lattice points. So given a sheet, you have to determine whether such cuts are possible, and if it is possible then in how many ways.

Input

The input file contains at most 800 lines of inputs.

Each line contains an integer N ($0<N<10^{14}$), which means that we have to cut an (NxN) sheet.

Input is terminated by a line containing a single zero. This zero should not be processed.

Output

For each line of input produce one line of output. This line contains the serial of output followed by a string or an integer. If it is not possible to cut the (NxN) sheet in the prescribed

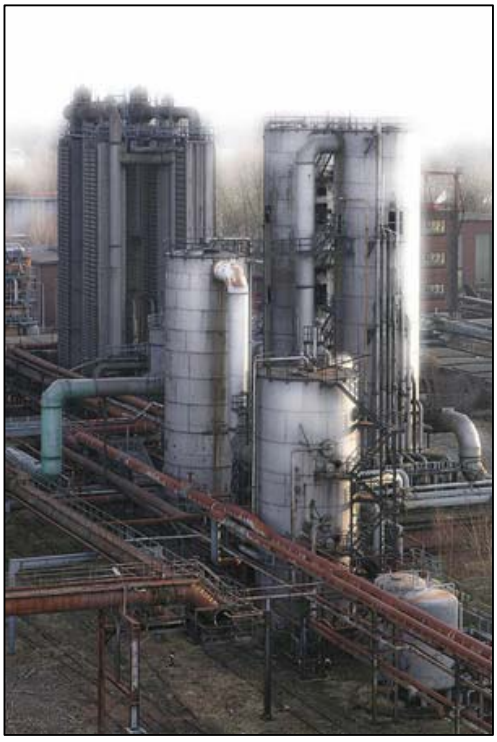
way, then print a line "`Impossible`", otherwise print an integer W. Here W denotes the number of ways such a cut is possible. Look at the output for sample input for details

Sample Input	Output for Sample Input
10 20 100 32 0	Case 1: Impossible Case 2: 8 Case 3: 72 Case 4: 24

E Chemical Plant

Input: Standard Input
Output: Standard Output

Dealing with chemicals is a risky job, particularly if it is reactive. Reactive chemicals can be dangerous if not stored in an appropriate environment. As a result, sophisticated instruments that can support real-time processing, is required while transferring a reactive chemical from one part of a chemical plant to another one. However, instruments that can ensure strictly real-time service, is too much costly & hence chemical plants use networks that provide real-time transfer with some tolerance.



Let us be a bit more elaborate. In our concerned chemical plant, there are V pits. Pits are places where chemicals can be stored. In order to reduce cost, they are not facilitated with reactive chemical handling environment. So, for every second the reactive chemical stays in some of these pits, they are decayed by 1 milligram. The pits are numbered from 1 to V. There is a safe storage attached to pit S. No time is required to move to the storage from pit S. There is some reactive chemical weighing W milligram in pit 1 at time 0. This chemical is to be transported to the safe storage. Note that, the safe storage door is opened exactly at time T. So, if the chemical reaches the storage before time T, you will have to accept decaying of the chemical whereas if it reaches after time T, it can not enter the safe storage. There are E transfer tubes in the network. The transfer tubes have suitable environment for storing reactive chemicals inside them. So, the chemicals are not decayed inside the tubes. Each transfer tube connects two pits with an entry point in pit s and an exit point in pit d. For each tube t, the entry point door is opened for any 1 second between time sst & sct (sst ≤ sct) but we don't exactly know in which second it is opened. Similarly, the exit point door is opened for any single second between time dst & dct (dst ≤ dct). As you don't know the exact timing of the doors' opening, if we want to move the chemical from pit s to pit d using tube t, it must be available at pit s no later than time sst. If the chemical reaches a particular pit x using tube y with exit point range dsy & dcy (dsy ≤ dcy) & leaves this pit using tube z with entry point range ssz & scz (ssz ≤ scz), the maximum possible decay in that pit is scz – dsy. Please note that, while choosing the tube to get out from a pit, you must always select a tube with it's opening interval beginning no sooner than the latest possible time for the chemical to enter that pit. For instance, in the 3rd sample test case, the chemical will always leave pit 2 using the 3rd edge even if it reaches pit 2 at time 15.

Write a program to find a way to transfer the chemical from pit 1 to the safe storage with maximum guaranteed weight left (in milligram) in the storage i.e. with minimum guaranteed decay. Please note that, the chemical can not have negative weight at any point.

Input

The input file contains multiple test cases. First line of each test case contains three integers, V (1≤V≤50,000), E (1≤E≤100,000) & W (1≤W≤2,000,000,000). The next line has two integers, S (1≤S≤V) & T (0≤T≤2,000,000,000)

Each of the following E lines describes a tube. A Tube t is described with 6 integers, s (1≤s≤V), d (1≤s≤V), sst, sct, dst, dct (0≤sst≤sct<dst≤dct≤2,000,000,000) where s, d, sst, sct, dst & dct holds the meaning as in problem statement.

The end of input is denoted with a case where V = E = W = 0. This case should not be processed.

Output

For each test case, print a single line of the form, "Plant C: L" where C is the test case number and L is the weight of the chemical available in the safe storage at time T.

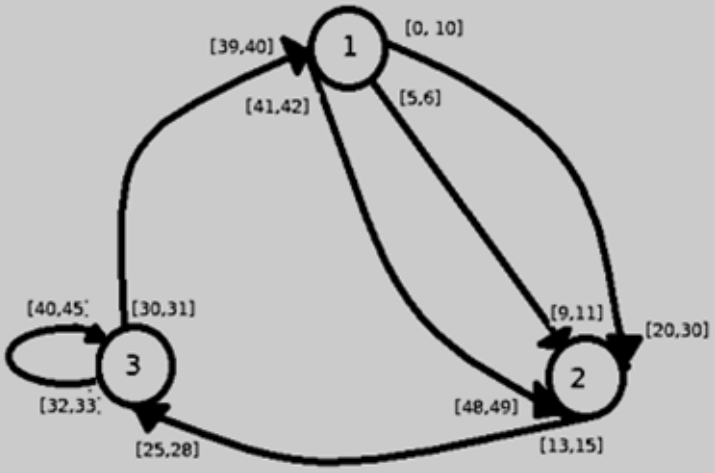
Sample Input

```
3 6 50
2 50
1 2 0 10 20 30
1 2 5 6 9 11
2 3 13 15 25 28
3 3 32 33 40 45
3 1 30 31 39 40
1 2 41 42 48 49
5 13 20
3 1000
3 3 41 41 999 1000
3 3 39 40 1000 1000
5 4 25 25 30 30
1 2 2 2 6 6
1 2 1 1 8 8
2 2 7 7 13 13
2 2 8 8 15 15
2 3 14 14 20 20
2 3 16 16 20 20
4 3 30 30 40 40
4 3 32 32 41 41
3 5 21 21 25 25
3 5 22 22 25 25
3 3 50
3 30
1 2 5 10 15 25
2 3 20 20 30 30
2 3 25 25 30 30
0 0 0
```

Output for Sample Input

```
Plant 1: 27
Plant 2: 15
Plant 3: 30
```

1st Sample Illustration:



The above diagram depicts the first case in sample. The chemical starts at time 0 in pit 1. Then goes to 2 by the ([5,6] [9,11]) edge. Minimum guaranteed decay at 1 is 6. Then 2 to 3 via the ([13,15] [25,28]) edge. Minimum guaranteed decay is 6. Then 3 to 1 through tube ([30,31] [39,40]). Again with 6 minimum guaranteed decay. Finally 1 to 2 by the ([41,42] [48,49]) edge. The minimum guaranteed decay is 3 at node 1 & 2 at the destination.

F Clicking Checkboxes

Input: Standard Input
Output: Standard Output

Creating a good and easy Graphical user interface is the heart of operating a computer in modern days. One important part of this interface is enabling users to give inputs using mouse and keyboard. Many different types of forms are there to take inputs from users. Such as (a) Text Box (b) Radio Button (c) Check Box (d) Combo Box etc. In this problem we will concentrate on check box.

Check boxes are used when a user needs to select more than one choice. Such a checkbox is shown in the figure on the left below. The problem with checkbox is that when someone has to choose n options or names he has to make n clicks, which can be very annoying when n is high. So the new software giant **toggle** is using a new type of checkbox, which shows good performance when number of boxes to be checked is high. In normal check box, a box is checked or unchecked if and only if it is clicked but in **toggle** checkbox when a box is clicked then all the boxes below it is also gets its status. That is if a box is originally not checked then if someone clicks this box then this box and all the boxes below it will be checked. Similarly if a box is checked and someone clicks on it then it will become unchecked and so will all the boxes below it. So the selection made on the left can be made by only five clicks (The boxes that has to be clicked are marked with red or dark square on the right figure) in **toggle** checkbox (We are calling naming new type of checkbox as "toggle checkbox").

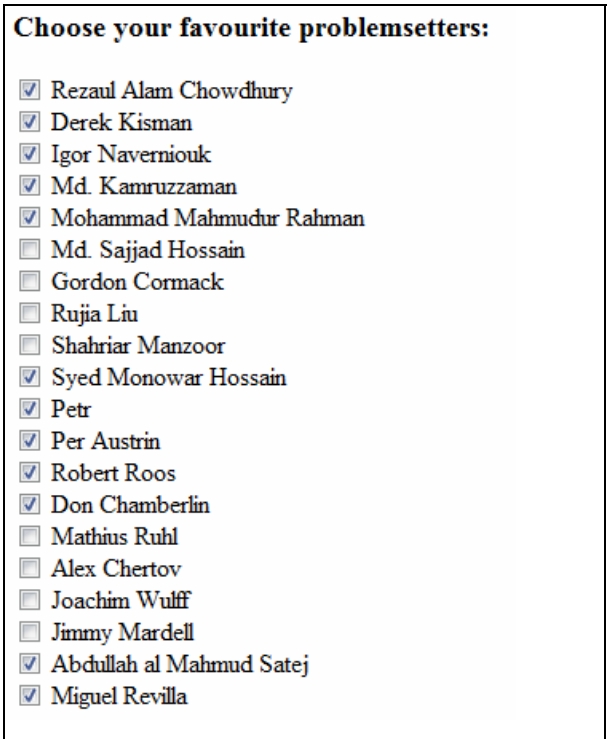


Figure 1: 12 clicks are required in ordinary checkbox

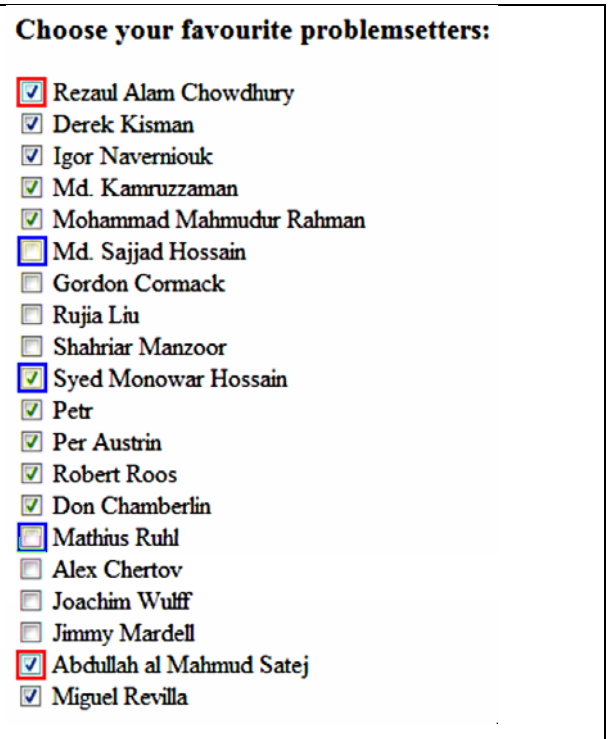


Figure 2: 5 clicks are required in toggle checkbox.

The researchers of toggle have shown that **toggle** checkboxes show better performance when at least 50% boxes are to be checked. For example for the selection shown in the above picture we needed 12 clicks to check 12 boxes in normal checkbox but only five clicks were required for **toggle** checkboxes. So in this situation **toggle boxes** show better performance.

In many other selection, **toggle** checkbox would also show strictly better performance. Your job is to count in how many selection patterns (where at least m boxes out of the n boxes in the checkbox are to be selected) toggle check box would show better performance and in how many selection patterns ordinary checkbox would show better performance.

Input

The input file contains 1000 lines of inputs. The description of each line is given below:

Each line contains two integers n ($0<n<64$) and m ($0 \le m \le n$). These two values indicate that you have to consider a checkbox with n boxes to check and consider only the input patterns where at least m of them are checked. Assume that initially all the boxes are unchecked.

Input is terminated by a line containing two zeroes. This line should be processed.

Output

For each line of input produce one line of output.

This line contains the serial of output followed by two integers TC and NC. Here TC is the total number of selection patterns where toggle checkbox requires less clicks and NC is the total number of selection patterns where ordinary or normal checkboxes require less clicks.

Sample Input	Output for Sample Input
5 3 10 3 0 0	Case 1: 9 3 Case 2: 370 419

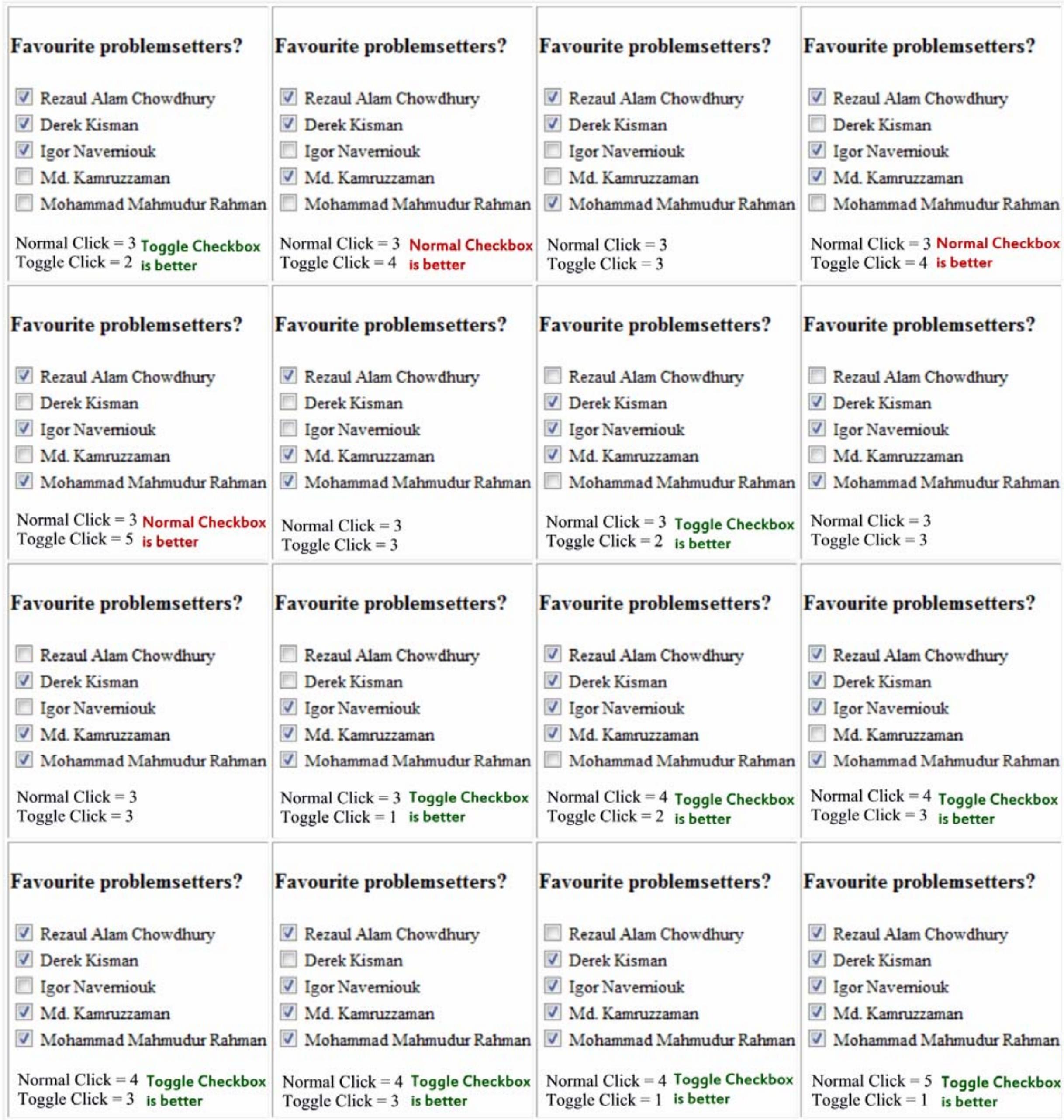


Illustration of first sample Input/Output:
When the total number of boxes that can be checked is five and at least three boxes has to be selected the total possible patterns for checking is $\frac{5!}{3!2!}+\frac{5!}{4!1!}+\frac{5!}{5!0!}=10+5+1=16$. Of these 16 patterns nine patterns are such that Toggle Checkbox shows better performance, three patterns are such that Normal Checkbox shows better performance while other four others are tied.

G

Magic Rings

Input: Standard Input

H Line Chart

Input: Standard Input
Output: Standard Output

ACRush is very famous in Supercoder. Supercoder is a professional company which arranges online algorithmic contests and rates peoples based on those contests. In Supercoder algorithm contest ranklist, ACRush is ranked third. Now a days he is doing some analysis on his rating history in Supercoder algorithm contest. In Supercoder, an algorithm contest is termed as a Single Round Tournament (SRT). After each SRT is finished, rating of a contestant is updated according to his/her relative performance. ACRush collected all these rating information, and using those he created a line chart.

To make things more clear, let us consider the following table as his rating info.

SRT	Rating
320	3
306	1
401	3
325	4
393	5
380	2

From this table, we see that his first SRT was SRT#306, and rating after that SRT was 1, so he marked point (1, 1) as r_1 in graph paper, his second SRT was SRT#320 and rating after that SRT was 3, so he marked (2, 3) as r_2, then he add r_1 with r_2 by a straight line and so on.

In general for his i^{th} SRT he marked point (i, rating after i^{th} SRT) by r_i.

After marking all the points he will add point r_i with r_{i-1} by straight lines, for all $1 < i <= N$, Where N is the total number of SRTs he played. For better idea look at figure 1:

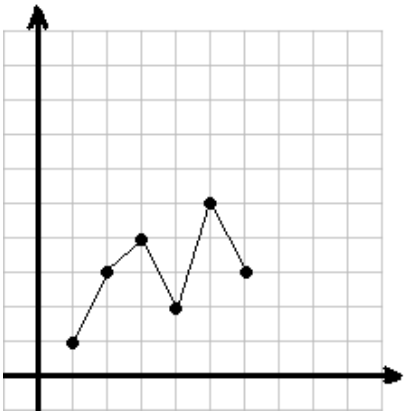


Fig 1: Line chart cosidering all SRTs

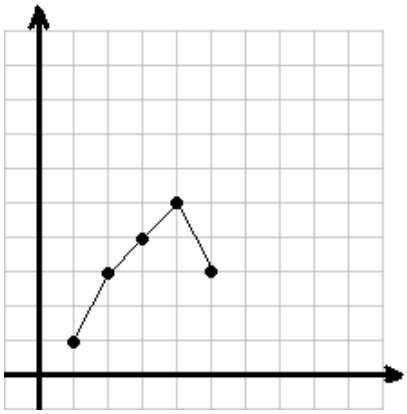


Fig 2: Line chart ignoring SRT #380

After drawing line chart, he became very interested about the number of peaks. There are two kinds of peaks in a line chart, 1) Upper Peak and 2) Lower Peak. Upper Peak is that point in a line chart whose previous and next point has smaller y coordinates and lower peak is that points in a line chart whose previous and next point has greater y coordinates. For example total number of peak in figure 1 is 3. Two of them upper peak, which are (3, 4) and (5, 5), and one of them is lower peak which is (4, 2).

ACRush observed that by ignoring SRT#380, his line chart will become like figure 2, in which number of peak is only 1. By observing this he became more curious. Now he wants to know, by ignoring 0 or more SRTs how many distinct line charts having K peaks is possible. ACRush calls these line charts "K-peak Line charts", in a K-peak line chart he doesn't allow two consecutive points to have same y coordinate.

Input

Input will start with an integer T (T ≤ 12), which indicates the number of test cases. Each case starts with a line having two integers N (1 ≤N ≤ 10000) and K (0 ≤ K ≤ 50). Each of the next N lines will contain two integers SRT (1 ≤ SRT ≤ 1000000000) and Rating (1 ≤ Rating ≤ 1000000000). All the SRT numbers will be distinct.

Output

For Each test case output a single Line "Case #: W", here # will be replaced by case number and W will be replaced by the number of distinct K-peak line charts modulo 1000000.

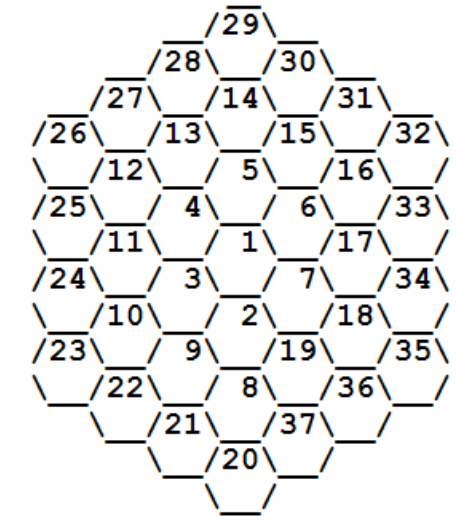
Sample Input	Output for Sample Input
3 6 1 320 3 306 1 401 3 325 4 393 5 380 2 4 1 101 3 102 2 103 2 104 4 3 0 102 2 101 1 103 3	Case 1: 20 Case 2: 1 Case 3: 8

I Mobile Towers

Input: Standard Input
Output: Standard Output

Byte Communication Inc. (BYTECOM) is establishing its cellular network in Byteland. From the first month of their establishment they are expanding their network very aggressively. Every month they are constructing some transmission tower. They are considering the following policy while establishing their networks.

1. BYTECOM have divided the Byteland into a logical hexagonal grid with N levels. Below is the example of a hexagonal grid with 4 levels.



First level is the center and it contains only cell 1. The number of cells in subsequent levels is 6,12,18,….. 2nd level contains cells from 2 to 7, 3rd level contains cells from 8-19, 4th level contains cells from 20-37,… In each of the first the lowest cell is numbered. Then the rest of the cells are numbered in clockwise order.

2. Each transmission tower covers exactly one cell. So they will not build multiple towers in the same cell.
3. There are 3 types of lines.
   - The lines parallel to the connecting lines of the center of cell 1 and cell 2 is TYPE1 line. For example cell 14 and cell 29 are in the same TYPE1 line. Same for the cell pairs (4, 28), (32, 35), (23, 25) etc.
   - The lines parallel to the connecting lines of the center of cell 1 and cell 3 is TYPE2 line. For example cell 15 and cell 11 are in the same TYPE2 line. Same for the cell pairs (4, 31), (10, 32), (16, 23) etc.
   - The lines parallel to the connecting lines of the center of cell 1 and cell 4 is TYPE3 line. For example cell 13 and cell 17 are in the same TYPE3 line. Same for the cell pairs (5, 27), (10, 37), (16, 28) etc.
   - To clarify more there is not any other type of line. For example cell 3 and cell 5 is not in any type of line. Same for the cell pairs (4, 6), (12, 29) etc.

   Every month BYTECOM will construct a number of cellular towers. But they have constraints for each the numbers of towers in each of the line type.
   - Every month they can construct unlimited number of towers in each of the TYPE1 lines.

- Every month they can construct at most 2 towers in each of the TYPE2 lines.
- Every month they can construct at most 3 towers in each of the TYPE3 lines.

4. Initially the construction cost of a tower in cell i is C_i units of money. But because of inflation it will increase by 1 unit of money every month. For example if C_5 is 10 then construction of a tower in cell 5 is 10 units of money in first month, 11 units of money in second month, 12 units of money in third months and so on.
5. On month i BYTECOM will construct M_i number of towers with following the policy above. Their plan is to minimize the cost of construction of these M_i towers. In each month they just want to minimize the cost of that month, they don't care for the following months.

Now given information of the cells and the monthly plans, help BYTECOM to minimize the cost of tower construction for each of the months.

Input

Input contains multiple numbers of test cases. First line contains T($1 \le T \le 10$) the number of test cases. Each test case consists of 3 lines. First line contains n($2 \le n \le 20$) and m($1 \le m \le 10$). n is the number of levels in cellular grid and m is the number of months for which BYTECOM is constructing cellular towers. Second line contains $3n^2-3n+1$ integers (the total number of cells) .The i'th integer is C_i ($1 \le C_i \le 1000$) denoting the cost of construction of a tower in cell i at first month. Third line contains m integers. The i'th integer is M_i ($1 \le M_i \le 50$) denoting the number of towers that need to be constructed on i'th month.

Output

For each test case output contains M+2 lines. First line is "Case i:" where i the number of test case. Each of the next M lines contains the cost for the i'th month in the following format.
"Month i: c unit of money"
Where c is the minimum construction cost for M_i towers in month i.

Also the input guarantee that for every month the set of M_i cells where building M_i towers costs c amount of money will be unique. No other set of unused cells will cost c or less amount of money in that month. An unused cell is the one where no tower has been built yet.
Also it will be always possible to build M_i number of towers in i'th month.

The last line is blank.

See the output for sample input for details.

Sample Input	Output for Sample Input
3 2 2 3 4 3 7 6 7 5 6 1 3 2 9 9 9 3 3 10 13 6 11 11 5 13 11 7 5 12 4 6 9 10 8 3 3 9 10 5 5 10 6 4 4 3 5 5 7 8 10 12 6 9 5 3 10 8 1	Case 1: Month 1: 28 unit of money Month 2: 8 unit of money Case 2: Month 1: 67 unit of money Month 2: 85 unit of money Case 3: Month 1: 49 unit of money Month 2: 76 unit of money Month 3: 11 unit of money

J Stopping Doom's Day

Input: Standard Input
Output: Standard Output

So! The time of the universe is up and it is the dooms day after five hours :-P, and you must stop it. But to do so you have to know the value of the following expression T:

$$T = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n} |(|i-j|*|j-k|*|k-l|*|l-m|*|m-i|)| \quad \% 10007$$

Because the secret code that will save the universe from being doomed have something to do with the value of the above expression for some value of n.

Input

The input file contains 1000 lines of inputs.

Each line contains a single integer n (0<n≤2000000000).

A line containing a single zero terminates input.

Output

For each line of input produce one line of output. This line contains the value of T.

Sample Input	Output for Sample Input
12 20 1001 0	2199 803 2390