The 2012 ACM ICPC Asia Regional Contest Dhaka Site

Sponsored by IBM

Hosted by Daffodil International University
Dhaka, Bangladesh

8th December 2012
You get 20 Pages
11 Problems
&
300 Minutes

acm International Collegiate Programming Contest

IBM event sponsor

Rules for ACM-ICPC 2011 Asia Regional Dhaka Site:

a) Solutions to problems submitted for judging are called runs. Each run is judged as accepted or rejected by the judge, and the team is notified of the results. Submitted codes should not contain team or University name and the file name should not have any white space.

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 while seated at the team desk. **But they cannot even talk with their team members when they are walking around the contest floor to have food or any other purpose.** Systems support staff may advise contestants on system-related problems such as explaining system error messages. **Coaches should not try to enter the contest area under any circumstances.**

e) While the contest is scheduled for a particular time length (five hours), the contest director 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 Contest Director** for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, distracting behavior oe communicating with other teams. The **external judges** will report to the **Judging Director** about distracting behavior of any team. **The external judges can also recommend penalizing a team with additional penalty minutes for their distracting behavior.**

g) Nine, ten or eleven 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 problems at least two will be solvable by a first year computer science student, another one 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 external judges. **The contestants are not allowed to communicate with any contestant (Even contestants of his own team) or coach while are outside the contest floor.**

i) Team can bring up to **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 and external judges, the contestants can have printouts of their codes for debugging purposes. **Passing of printed codes to other teams is strictly prohibited.**

k) The decision of the judges is final.

l) Teams should inform the volunteers if they don't get reply from the judges within 10 minutes of submission. Volunteers will inform the External Judges and the external judge will take further action. Teams should also notify the volunteers if they cannot log in into the PC^2 system. This sort of complains will not be entertained after the contest.

m) If you want to assume that judge data is weaker than what is stated, then do it at your own risk :).

A Divisibility

Input: Standard Input
Output: Standard Output

You are in the system of **N**-dimensional infinite hyper-grid with each hyper cell having an integer. In an N-dimensional grid the co-ordinates of a cell are denoted as **(X_1, X_2, ..., X_N)**. Any hyper cell with at least one negative co-ordinate contains the value **0** (zero). The origin hyper cell (the one with all zero co-ordinates) contains the value **1**. The value of a hyper cell with co-ordinate **(X_1, X_2, ..., X_N)** (with all non-negative **X_i**) is the sum of the values in **N** hyper cells with co-ordinates **(X_1-1, X_2, ..., X_N)**, **(X_1, X_2-1, ..., X_N)**, ..., **(X_1, X_2, ..., X_N-1)**. You are given the starting and ending co-ordinate of a sub-hypercube. You need to compute how many hyper cells in this sub hypercube contain an integer **not** divisible by a given prime **P**.

Input

First line of the input contains **T (0 < T < 51)** the number of test cases. Each test case starts with a line containing **N (0 < N < 8)** the dimension of the hypercube and the prime **P (1 < P < 20)**. The second line contains **N** integers denoting the co-ordinate of the starting cell of the hypercube. The third line contains **N** integers denoting the co-ordinate of the ending cell of the hypercube. All the co-ordinates will be non negative integers with at most **15** digits.

Output

For each test case, print the serial of output followed by the number of hyper cells in the given sub hypercube that contains an integer not divisible by a given prime **P**. Since the result can be too big so output the result modulo **1000000009**. Look at the output for sample input for details.

Sample Input

```
3
3 2
4 0 4
7 9 8
4 3
0 3 0 2
6 8 1 5
5 7
1 2 3 4 5
11 12 13 14 15
```

Output for Sample Input

```
Case 1: 9
Case 2: 17
Case 3: 2515
```

Problemsetter: Abdullah al Mahmud, Special Thanks: Derek Kisman

B

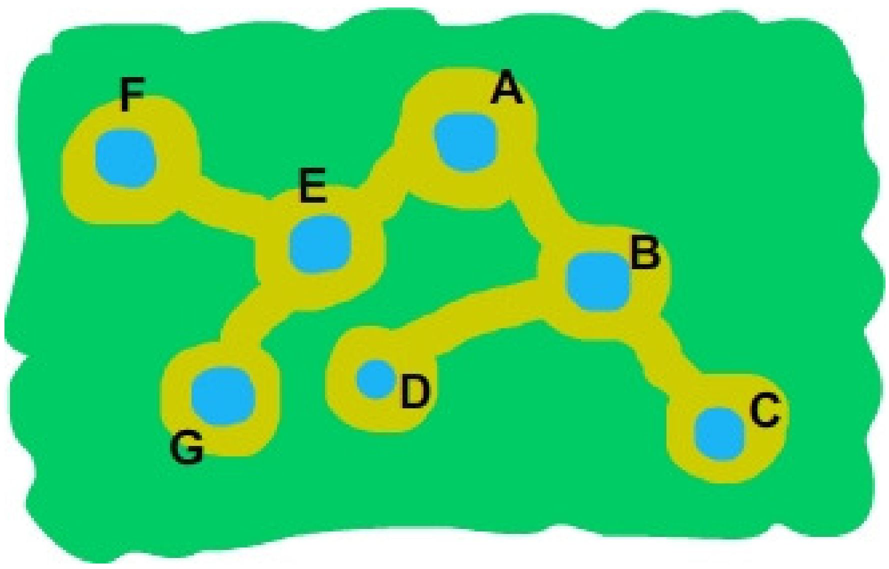
Wedding of Sultan

Input: Standard Input
Output: Standard Output

As usual Sultan Mahmud is very busy. He works days and nights at office. If you ask him, “Sultan, which day of the week is this?” He will look at you for a while and say, “I think I have **3** more days till deadline!” But one day the scenario changed after receiving a call. He usually ignores phone calls from everyone (even from his fiancée) but this time he couldn’t ignore because of the importance of the person! This person was his to-be mother-in-law. So he received the call and heard, “Son, only **30** days left of your wedding ceremony, so I am sending a tailor for the measurement for your suit.” Sultan now remembered, he is about to get married but looking at thyself, he got surprised! When did he get so fat! “Umm.. Mom can it be arranged **10** days later?” He wants to buy some time so that he can exercise and lose extra weight. So he immediately went out with his bicycle to the large garden beside his house.

There are several trails in the garden. A trail starts from one water sprinkler to another and the sprinkler are marked by distinct letters from **‘A’** to **‘Z’**. The trails are designed in such a way that from the sprinkler at entrance you can go to any other sprinkler using exactly one path if you do not traverse a trail more than once.

While traversing the trails with his cycle, Sultan notes the names of the sprinklers in his notepad. He will write down the name of a sprinkler if he enters the sprinkler for the first time or leaves this sprinkler for the last time. And not surprisingly, geek Sultan follows a peculiar method to ensure that he visits all the trails of the garden. When he comes to a sprinkler he looks for a trail which he has not traversed yet. If he finds such trail, he follows that one. Otherwise, he uses the trail that he used to come here for the first time except if it’s the entrance he stops exercising. He always starts from the entrance and guess what, his peculiar strategy always guarantees to finish him at the entrance and all the trails are also visited.



For example, in the above garden the main entrance is at **A**. So Sultan will start from **A**. When Sultan is at **A**, he can choose either of the trails. Say he chooses the trail leading to **E**. Then he can choose the trail to **G** or trail to **F**. Say he chooses **F**. Now he does not have any unvisited trail from **F**, so he will go back to **E**. Now he must choose trail to **G** and then similarly will come back to **E** and back to **A**. Then he will go towards **B**. Now he again has two choices. He can go to **C** or **D**, say he goes to **C**, then he will be back to **B**, then will go to **D**, and hence back to **B** and also back to **A** thus finishing his exercise. So after his exercise you will find: **AEFFGGEBDDCCBA** in his notepad. Can you find the number of trails attached to the sprinklers just looking at the sequence written in the notepad?

Input

First line of the test file contains a positive integer **T** (**T** ≤ **100**) denoting the number of test cases.

Hence follows **T** lines, each containing a valid sequence of sprinkler names. A sprinkler name will always be capital Latin letter (**'A', 'B' … 'Z'**). You may assume that there will be at least two sprinklers in garden, otherwise there would have been no meaning of exercise right?

Output

For each case output the case number in the first line, followed by the number of trails for each sprinkler. First print the sprinkler name followed by the count of trails. These lines should be in lexicographical order of sprinkler name. Note that, you should not print about a sprinkler which is not present in the garden. Look at the sample input output for more specific format of input output.

Sample Input

```
2
AEFFGGEBDDCCBA
ZAABBZ
```

Output for Sample Input

```
Case 1
A = 2
B = 3
C = 1
D = 1
E = 3
F = 1
G = 1
Case 2
A = 1
B = 1
Z = 2
```

Problemsetter: Md. Mahbubul Hasan, Special Thanks: Jane Alam Jan

C

Memory Overflow

Input: Standard Input
Output: Standard Output

The Great Sultan Mahbub is having some serious issues with his memory. His busy days filled with great works and surrounded by even greater people have brought him to a situation where he has become quite forgetful. For example, he often forgets trivial things like the size of his suit, his weight, small grammatical issues related to gender & number, the address of his in-laws house, how to ride a cycle, deadlines, the day of the week, the name of the guy who forgot his wedding day and even the date of his own wedding (thus spending the day writing alternate solutions to ICPC problems). But when his father-in-law captured him to his in-laws house and he failed to recognize his mother-in-law it became a fiasco. And after some rather presumable events following that debacle, the detail of which does not seem really safe to mention, he decided that the matter has become pressing enough for his attention. His physician is startled by this weird problem and decides to collect statistical data to begin with.

During the examination period consisting **n** consecutive days, the Sultan meets a single person everyday. He only recognizes the person if he has met him in the last **k** days (excluding today of course). You need to count the number of days (among these **n** days) he manages to recognize the people he meets. You can assume that before these **n** consecutive days he did not meet any person.

Input

The input begins with a number **t** **($1 \le t \le 100$)**, the number of test cases. Each of the following lines contains a case. A case begins with **n** **($1 \le n \le 500$)** & **k** **($1 \le k \le 500$)**. A list of **n** names follows. All names consist of a single uppercase letter and names are unique. They are given in the order of which Sultan meets them during the investigation. There won't be any invalid character or space between any two names.

Output

For each test case produce a line of the form "**Case X: Y**". **X** is the serial number of the test case while **Y** is the number of people Sultan recognizes.

Sample Input

```
3
6 2 SULTAN
6 1 MAHBUB
6 2 MAHBUB
```

Output for Sample Input

```
Case 1: 0
Case 2: 0
Case 3: 1
```

Illustration of Third sample: **<u>Day 1:</u>** Sultan remembers nobody, meets **'M'**. Does not recognize. **<u>Day 2:</u>** Remembers only **'M'** but meets **'A'**. Does not recognize again. **<u>Day 3:</u>** Now remembers **'M'** & **'A'**. Meets **'H'**. Recognition count remains 0. **<u>Day 4:</u>** Forgets 'M', remembers **'A'** & **'H'**. Meets **'B'**. Still nothing happens. **<u>Day 5:</u>** Forgets **'A'**, remembers **'H'** & **'B'**. Meets **'U'**. No luck yet. **<u>Day 6:</u>** Forgets **'H'**, remembers **'B'** & **'U'**. Meets **'B'** again and recognizes this time making the recognition count **1**.

Problemsetter: Md. Mahmudur Rahman, Special Thanks: Shahriar Manzoor

D

Laptop Chargers

Input: Standard Input
Output: Standard Output

ABC has set many problems for programming contests but has bought even more laptops in his life. Fortunately, all his laptops are still working but some of his chargers are not working. If he has **N** laptops (i) Minimum how many chargers does he need to run all the laptops forever? (ii) Maximum how long will he be able to run all his laptops using **M (M<N)** chargers? You need to assume the following things for simplicity:

(i) The battery of each laptop has a charge capacity **C** (Expressed in mAh) (ii) The battery discharge amount is always strictly linear with time (eg: It looses fixed mAh charge in every second) and the fully charged battery of the laptop completely discharges in time **T** (Expressed in seconds) (iii) The charge that the charger adds with the battery of a laptop is always linear with time as well. (iv) All chargers are identical and any laptop is compatible with any charger. (v) The charging rate of a charger is strictly greater than discharging rate of any laptop. (vi) Zero time is needed to shift a charger from one laptop to another. If needed this shifting can be done indefinite amount of times. (v) Exactly one charger can be used to charge a laptop.

Input

The input file contains around **1000** sets of input. The description of each set is given below.

Each test case starts with two integer **N** $(0 < N < 101)$ and **Q** $(0 < Q \leq 10)$. Here **N** is the total number of laptops owned by ABC and **Q** is the total no of query. The 2nd line describes the chargers (All chargers are identical). This line contains an integer **Chps** $(1 \leq Chps \leq 3)$ which denotes the amount of charge (expressed in mAh) the charger adds with the battery in a second. Each of the next **N** lines describes the status of one laptop. Each laptop is described with three integers **C**, **T** and **R**. Here **C** $(2000 \leq C \leq 10000)$ is the capacity of Battery (Expressed in mAh), **T** $(3000 \leq T \leq 28800)$ is the time required for the fully charged battery to discharge (Expressed in seconds), **R** $(0 < R \leq C)$ is the charge remaining in the laptop battery (Expressed in mAh) at the beginning. Each of the next **Q** lines contains an integer **M** which describes number of chargers ABC has in his possession.

Input is terminated by a line containing two zeroes.

Output

For each set of input you should produce **Q+2** lines of outputs. First line is the serial of output and second line contains an integer M_{min} which denotes minimum how many charges are needed to run all the laptops indefinitely. You can assume that input will be such that small precision error will not change the value of M_{min}. Each of the next **Q** lines contains a floating-point number **T** (rounded to three digits after the decimal point) which denotes maximum how long (In seconds) the **N** laptops can run with **M** chargers. But if with **M** chargers the laptops run more than **100000** seconds output **-1.000** instead. Look at the output for sample input for details. For the value of **T**, an absolute error of less than **0.02** will be ignored.

Sample Input

```
4 2
2
3269 6150 3117
4135 5839 2770
3377 5552 779
7452 17787 2924
2
2
0 0
```

Output for Sample Input

```
Case 1:
2
-1.000
-1.000
```

Problemsetter: Shahriar Manzoor, Special Thanks: Derek Kisman, Rujia Liu

E

Poker End Games

Input: Standard Input
Output: Standard Output

Alice and Bob loves playing poker with their friends. Unfortunately, they play poker way better than their friends. So, almost always they are the last two players to play. Two of them can play for a long time and it bores their friends. So they changed the rule little bit and decided that both of them will go all in every round. Now, Alice is wondering what is the expected length of the game, and what is the probability that she will win the game.

Let's say at the beginning of round **i**, Alice has $\mathbf{a_i}$ Taka (Currency of Bangladesh) and Bob has $\mathbf{b_i}$ Taka. and $\mathbf{c_i}$ is the minimum of $\mathbf{a_i}$ and $\mathbf{b_i}$. Alice and Bob are equally likely to win the round. If Alice wins, she gets $\mathbf{c_i}$ Taka from Bob, otherwise Bob gets $\mathbf{c_i}$ Taka from her. Game ends when one of them has **0** (Zero) Taka and obviously the person with **0** taka loses.

Given that the initial amount Alice has is $\mathbf{a_0}$ and the initial amount that Bob has is $\mathbf{b_0}$, you have to find the probability that Alice is going to win and expected number of rounds the game is played.

Input

Input file starts with a number **T** ($\mathbf{0 < T \leq 100}$). **T** test cases follow. The input for each test case is contained in a single line and it consists two integers **a** and **b** ($\mathbf{0 < a, b \leq 100}$).

Output

For each case, print the case number followed by the expected number of rounds and the probability that Alice will win. Print both result rounded to **6** digits after the decimal point. For both these values errors less than $\mathbf{10^{-5}}$ will be ignored.

Sample Input

```
2
1 1
2 1
```

Output for Sample Input

```
Case 1: 1.000000 0.500000
Case 2: 2.000000 0.666667
```

Problemsetter: Tanaeem M. Moosa, Special Thanks: Md. Mahbubul Hasan

F Overlapping Characters

Input: Standard Input
Output: Standard Output

All the uppercase English letters and decimal digits (total **36** characters) can be written in **(16*43)** grids (**16** rows and **43** columns) using dots ('.') and asterisks ('*'). Some such characters are shown below. As you can see that the empty pixels are marked with '.'s and black pixels are marked with '*'s.

```
..*******..................................
.********..................................
*********..................................
*********..................................
...******..................................
...******..................................
...******..................................
...******..................................
...******..................................
...******..................................
...******..................................
...******..................................
************...............................
************...............................
************...............................
************...............................
```

Grid representation of '1'

```
.....*********.............................
...*************...........................
.*****************.........................
*******************........................
*******.....*******........................
*******.....*******........................
.*****************.........................
..***************..........................
.*****************.........................
*******.....*******........................
*******.....*******........................
*******.....*******........................
*******************........................
.*****************.........................
...*************...........................
.....*********.............................
```

Grid representation of '8'

```
...............***.........................
..............*****........................
.............*******.......................
............*********......................
...........***********.....................
..........*************....................
.........*******.*******...................
........*******...*******..................
.......*******.....*******.................
......*********************................
.....***********************...............
....*************************..............
...*******.............*******.............
..*******...............*******............
.*******.................*******...........
*******...................*******..........
```

Grid representation of 'A'

```
**********.................................
**********.................................
**********.................................
**********.................................
..******...................................
..******...................................
..******...................................
..******...................................
..******...................................
..******...................................
..******...................................
..******...................................
**********.................................
**********.................................
**********.................................
**********.................................
```

Grid representation of 'I'

```
*****************..........................
******************.........................
*******************........................
********.....*******.......................
..******.....*******.......................
..******.....*******.......................
..*****************........................
..****************.........................
..**************...........................
..******...................................
..******...................................
..******...................................
**********.................................
**********.................................
**********.................................
**********.................................
```

Grid representation of 'P'

```
********...........................********
********...........................********
********...........................********
********...........................********
.*******...........*****...........*******.
..*******.........*******.........*******..
...*******.......*********.......*******...
....*******.....***********.....*******....
.....*******...*************...*******.....
......*******.*******.*******.*******......
.......*************...*************.......
........***********.....***********........
.........*********.......*********.........
..........*******.........*******..........
...........*****...........*****...........
............***.............***............
```

Grid representation of 'W'

```
..************.............................
.***************...........................
******************.........................
*******************........................
*********...*******........................
*********...*******........................
.*****************.........................
..***************..........................
.*****************.........................
*********...*******........................
*********...*******........................
*********...*******........................
*******************........................
******************.........................
****************...........................
**************.............................
```

Top view of the pile when grid for character 8 is put on grid for character 1.

```
*****************..................********
******************.................********
*******************................********
********.....*******...............********
.*******.....***********...........*******.
..*******....************.........*******..
..************************.......*******...
..*************************.....*******....
..**************************...*******.....
..***********.*******.*******.*******......
..******************...*************.......
..*****************.....***********........
******************.......*********.........
*****************.........*******..........
**********.*****...........*****...........
**********..***.............***............
```

Top view of the pile when grid for character P is put on grid for character W.

When these characters are placed one over another (without scaling, translation or rotation) it is very difficult to judge from above how many characters are there in the pile. Sometimes it is also very difficult to say which characters are in the pile but sometimes it is very easy as you can tell whether a character is present in the pile just by checking a single pixel (if the pixel is black the character is present, otherwise absent). Suppose out of the **36** characters you have specific grid representation of specific **M** characters with you. From those **M** character grids your friend can choose any number of grids and place them in any order to form a pile and then ask you to guess which characters are present there. But you are clever and you know that there are certain characters whose presence in the pile can be detected just by checking a single pixel (regardless of whatever your friend does). Given the characters in hand your job is to identify the characters whose presence can be detected by checking only a single pixel.

Input

There is only a single set of input.

First line of the input file contains two integers **N** **(1 ≤ N ≤ 36)** and **Q** **(1 ≤ Q ≤ 1000)**. Here **N** is the total number of character grids to consider for all the queries of this problem and **Q** denotes the total number of queries. The next line contains a string consisting exactly **N** distinct uppercase alpha-numerals. This string denotes the order in which the description alpha-numeral grids will appear in the input. The next **17N** lines describe the grid for **N** characters. Each character-grid is described in exactly **16** lines but there is a line containing only empty pixels following the description of each character.

Each of the next **Q** lines a string of **M** **(1 ≤ M ≤ 18)** characters which describes the set of characters from which your friend can choose any number of character grids and place on a pile in any order. All the characters in this query string will be distinct.

Output

For each query produce one line of output.

This line contains the serial of output followed by a string containing only **'N'** or **'Y'** (one for each character in the input query string). The printed character should be **'Y'** if the presence of the corresponding input character grid can be identified by checking whether a single pixel is black. **'N'** should be printed otherwise. (Note that sometimes if a single pixel is empty, the presence of a character can certainly be detected. But we are not considering that issue in this problem)

Sample Input

```
3 2
ABC
...............***.........................
..............*****........................
.............*******.......................
............*********......................
...........***********.....................
..........*************....................
.........*******.*******...................
........*******...*******..................
.......*******.....*******.................
......*********************................
.....***********************...............
....*************************..............
...*******.............*******.............
..*******...............*******............
.*******.................*******...........
*******...................*******..........
...........................................
*****************..........................
******************.........................
*******************........................
********.....*******.......................
..******.....*******.......................
..******.....*******.......................
..*****************........................
..****************.........................
..*****************........................
..******.....*******.......................
..******.....*******.......................
..******.....*******.......................
********************.......................
*******************........................
******************.........................
*****************..........................
...........................................
........*************......................
.....****************......................
...******************......................
..*******************......................
.*******.......******......................
*******....................................
*******....................................
*******....................................
*******....................................
*******....................................
*******....................................
.*******.......******......................
..*******************......................
...******************......................
.....****************......................
........*************......................
...........................................
ABC
AB
```

Output for Sample Input

```
Query 1: YYY
Query 2: YY
```

Problemsetter: Shahriar Manzoor, Special Thanks: Monirul Hasan

G

Reduce the Maintenance Cost

Input: Standard Input
Output: Standard Output

There are **n** towns (numbered from **1** to **n**) in a strange city and the maintenance costs of the towns are not necessarily same. There are bi-directional roads in the city and a road connects two towns. The merchants use the roads to trade amongst towns. If a road becomes unusable for some reason, it may affect the trading between some towns.

So, the mayor of the city has planned that every road will be maintained by one of its connecting towns. That means if a road connects town **a** and **b**, then either **a** or **b** (not both) will be given the responsibility to maintain the road.

The cost of maintaining a road is = **N*L**
Here **N** = The number of town pairs that cannot trade if this road is damaged
L = the length of the road

If a town is given the responsibility to maintain some roads then the maintenance cost of the town will be increased by the summation of maintenance costs of the roads it'd be maintaining. Now, you are given the information of the towns, your task is to assign the roads to towns such that the maximum maintenance cost of a town is as small as possible.

Input

Input starts with an integer **T** (**T ≤ 30**), denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n** (**2 ≤ n ≤ 10000**) and **m** (**0 ≤ m ≤ 20000**), where **n** denotes the number of towns and **m** denotes the number of roads respectively. The next line contains **n** space separated integers between **1** and **10000** (inclusive), where the **i-th** integer denotes the maintenance cost of the **ith** town. Each of the next **m** lines contains three integers **u v w** (**1 ≤ u, v ≤ n, 1 ≤ w ≤ 10000, u != v**), denoting that there is a road between town **u** and **v** whose length is **w**. There is at most one road between two towns.

Output

For each case, print the case number and the result.

Sample Input

```
3

2 1
5 10
1 2 10

6 6
10 20 30 40 50 60
1 2 1
2 3 1
1 3 1
1 4 6
1 5 6
4 6 2

3 1
10 20 30
2 3 10
```

Output for Sample Input

```
Case 1: 15
Case 2: 80
Case 3: 30
```

Note

Dataset is huge, use faster I/O methods.

For case **2**, if the road between **1** and **4** goes down the town pairs that will be affected are: **(1, 4), (1, 6), (2, 4), (2, 6), (3, 4), (3, 6), (4, 5), (5, 6)**.

Problemsetter: Jane Alam Jan, Special Thanks: Md. Mahbubul Hasan

H Team Mathematics Olympiad

Input: Standard Input
Output: Standard Output

Team Mathematics Olympiad is an interesting competition in which a team of n people will answer m questions. Here is a typical (but simplified) problemset:

Question 1. What is the sum of **123** and **987**?
Question 2. Let **x** be the answer of the previous question, how many digit does **x** have (in decimal form)?
Question 3. What is the smallest **6**-digit prime number?
Question 4. How many positive factors does **100!** have?
Question 5. Let **x** be the answer of the previous question, what is the last digit of x^x?

You see, some of the questions make use of the previous question's answer. That means, if your team's answer for the previous question was incorrect, then virtually you have no chance answering this question correctly (i.e. the probability of answering correctly is zero).

The good news is that you can take a look at the problemset and make some quick assessment before you start.

After that, you get a matrix **P**, in which $\mathbf{P_{i,j}}$ is the probability that the **i**-th person answer the **j**-th question correctly, given the correct answer to the previous question (if there is).

The bad news is that after the quick assessment, the problemset is taken away, and you're asked to answer the questions in order (i.e. answer question **1** first, then question **2**, **3**... You have to answer each question, if you know you have no chance doing correctly). Before answering each question, your team has to choose one person, then he enters a secret room, read the question, submit his answer and leave the room. Shortly after that, you'll be immediately informed whether the answer is correct, and your team prepare for the next question.

Anyone can go for any question, but the work assignment should be balanced. Let $\mathbf{C_i}$ be the total number of question that the i-th person actually answered during the competition, then $\mathbf{max(C_i)-min(C_i)}$ should not be more than **1**.

Unfortunately, the questions are very lengthy (unlike the simplified problemset above), so nobody can remember enough details to work on a question before going to the secret room.

Similarly, when working on a question in the secret room, nobody can remember the previous question. So if he already knew the previous answer was incorrect, he'll have no chance of answering the current question correctly if it is dependent on the answer of previous question.

Find out a strategy to maximize the expected number of correctly answered questions. Note that the strategy doesn't have to be static: you can make different assignment on different situations.

Input

The first line contains the number of test cases **T**$(1 \le T \le 100)$. The first line of each test case contains two integers **n** and **m** $(2 \le n \le 5,\ n \le m \le 30)$, the number of team members and the number of questions.

The next line begins with an integer **k** $(0 \le k \le m-1)$, the number of questions that needs the answer to the previous question. Then **k** different integers follows, the list of questions (questions are numbered from **1** to **n**).

The integers are sorted in increasing order, and does not contain **1** (the first question doesn't have "previous question"). Each of the next **n** lines contains **m** real numbers between **0** and **1**, the **j**-th number in the **i**-th line is $P_{i,j}$.

Output

For each test case, print the serial of output followed by the maximal expected number of correctly answered questions (if you follow the best strategy), to four digits after the decimal point. Look at the output for sample input for details. You can safely assume that errors not exceeding 10^{-5} will be ignored.

Sample Input

```
2
2 2
1 2
0.8 0.1
0.3 0.9
2 4
2 2 4
0.5 0.1 0.5 0.1
0.5 0.9 0.5 0.9
```

Output for Sample Input

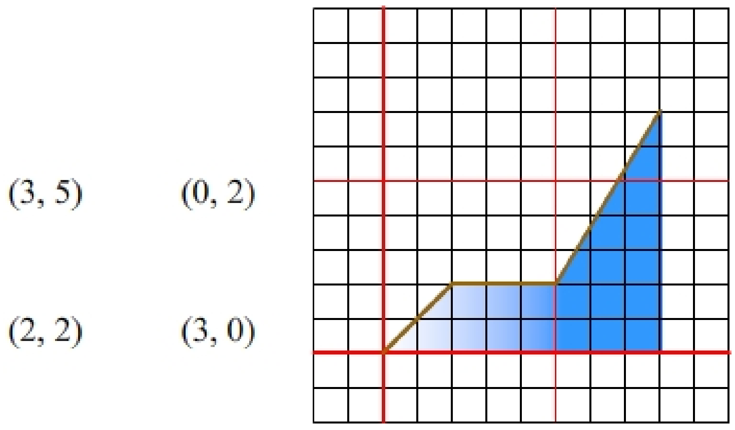
```
Case 1: 1.5200
Case 2: 1.9000
```

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan

I — Learning Vector

Input: Standard Input
Output: Standard Output

Vector is a very useful mathematical tool. It sometimes makes all the calculations very easy. So I decided to teach my **7** year old son this amazing tool. Don't be surprised, he is a very talented young child. He already understands coordinate, drawing line in grids etc. So when I say him to draw a vector **(4, 3)** from a point **(2, 7)** he draws a line from **(2, 7)** to **(6, 10)**. One day I decided to teach him the concept of area too. I gave him **N** vectors and told him to draw any **K** distinct vectors among those **N** vectors one after one. And also told him that I will give him cake of the size of the area he can bind by the polyline and **x**-axis. Surprisingly my kid is yet to capture the concept of negative number. So I decided to give him non-negative vectors only, that is both the **x** and **y** components of the vector will be non-negative. If you are wondering what happened after that, let me show you what he had drawn.



I gave him four vectors **(3, 5)**, **(0, 2)**, **(2, 2)** and **(3, 0)** and told him to draw **3** of them one after one. So he drew **(2, 2)**, **(3, 0)** and **(3, 5)** in this order shown in the diagram above. (He first drew **(2,2)** vector from **(0, 0)** to **(2, 2)**; then he drew **(3, 0)** from **(2, 2)** to **(5, 2)** and finally **(3, 5)** from **(5, 2)** to **(8, 7)**) These **3** vectors and x-axis bind area of **21.5**. Is it the maximum? That is your task.

Input

First line of the test case contains a single positive integer **T** which is at most **110**. Then there follows **T** test cases.

First line of a test case contains two positive integers **N** and **K** **($1 \le K \le N \le 50$)**. Then **N** lines follow, each contains description of a **2D** vector **(x, y)** (**$0 \le x, y \le 50$**). The meaning of **N** and **K** are given in the problem statement.

Output

For each test case print the test case number followed by the maximum area covered by the polyline. To avoid floating point calculation output **twice of the maximum area**. To be more clear, please follow the sample input output.

Sample Input

```
2
4 3
3 5
0 2
2 2
3 0
4 2
3 5
0 2
2 2
3 0
```

Output for Sample Input

```
Case 1: 81
Case 2: 45
```

Problemsetter: Md. Mahbubul Hasan, Special Thanks: Sohel Hafiz

J

Guards II

Input: Standard Input
Output: Standard Output

This ICPC will take place in a huge hall room which can be divided into **N x M** square cells. That's why some volunteers will guard this room. But each of the border cells must be guarded by at least one volunteer. And in a single cell at most one volunteer can be placed. Now volunteers can watch other cells vertically or horizontally (All cells that are in the same row or in the same column).

So we can consider that, there are **N** rows and **M** columns in the room. A volunteer at cell **(r, c)** (i.e. cell of **r**th row and **c**th column) can guard all the cells of **r**th row and **c**th column. A cell is border cell if it is in **1**st row or **N**th row or it is in **1**st column or **M**th column.

We have **K** volunteers. We must place exactly **K** volunteers. You have to determine, in how many ways we can choose **K** cells, so that each of the border cells will be guarded by volunteers if we place them in those cells.

Input

Input starts with a positive integer **T** (**T ≤ 20000**), which indicates the number of test cases. Each of the next **T** lines will contain three integers **N, M** and **K** (**1 ≤ N, M, K ≤ 100**) representing one test case.

Output

For each test case, output a single line in the form **"Case #: R"**, where # will be replaced by the case number and **R** will be replaced by the number of ways we can place **K** guards. This number can be very large, so output it **modulo (10^9 + 7).**

Sample Input

```
4
10 10 2
5 6 1
2 2 3
2 2 5
```

Output for Sample Input

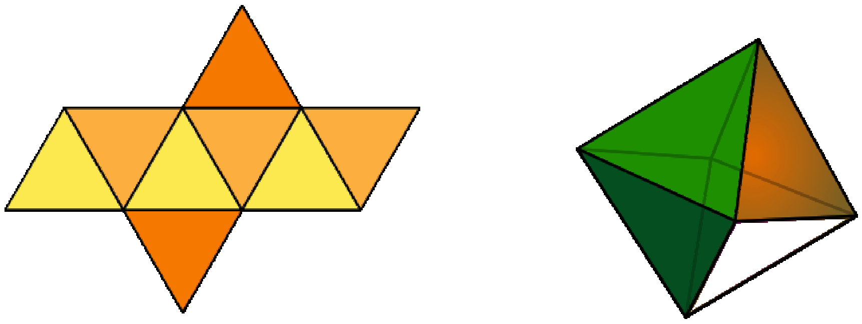
```
Case 1: 2
Case 2: 0
Case 3: 4
Case 4: 0
```

Problemsetter: Arifuzzaman Arif and Derek Kisman

K Beauty of Regular Polyhedron

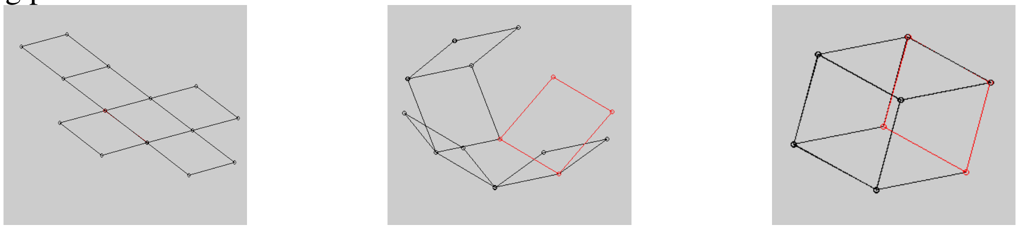
Input: Standard Input
Output: Standard Output

It is quite easy to draw regular polygons but what about regular polyhedron? If you are allowed to cut papers and join them to form regular polyhedron it would be easy task, but what if I tell you to fold papers to form polyhedron? Let me clear you by a picture excerpt from Wikipedia.



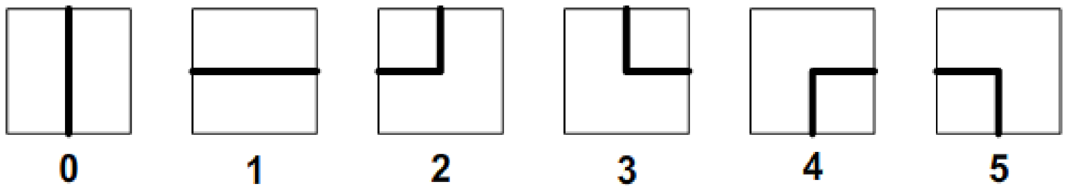
Left side paper can be folded to form an octahedron. If you don't know how an octahedron looks like, it is shown in the right side.

However you don't need to be scared seeing the octahedron. We will work with the simplest of all polyhedron, regular hexahedron. Actually regular hexahedron is formal name of cube. Let's look at the following pictures:

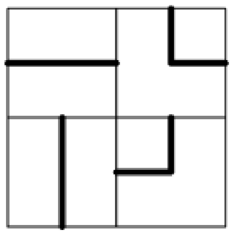


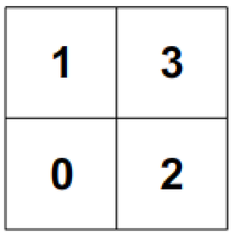
Left side paper can be folded to a cube. There are many other shapes of six connected squares which can be folded to form a cube.

In this problem you will be given a grid. It will be a R*C rectangle which is divided into unit squares. A square will look like one of the following:

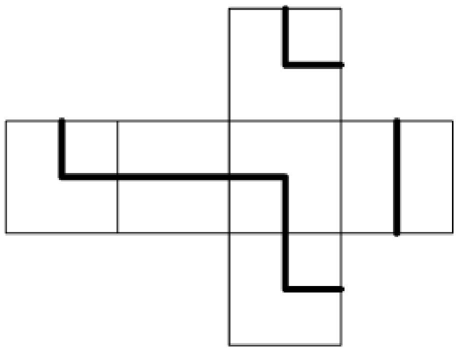


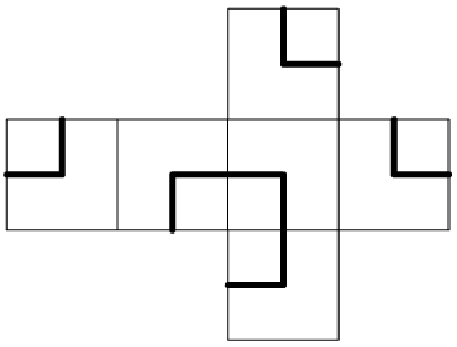
So a 2*2 grid may look the left diagram below (which as input to your program will be given as right one. You will find the meaning of the numbers in the above picture):





In how many ways can you cut 6 connected squares from the given R*C grid so that when a cube is formed folding them, the lines on the cube surfaces form a single loop? For example, if we cut out the left portion from a grid in the diagram below it will form a valid cube, but if we cut out the right portion, we can fold it to form a cube but the lines on the surfaces will not form a single loop. So it is not valid.





Note that, the lines on the grid are in both sides and the lines in both the sides are same. So you don't need to worry which side of the grid square is inside cube.

Input

First line of the test case contains a positive integer **T** (**T ≤ 100**). Hence follows **T** test cases. Each test case starts with **2** positive integer **R** and **C** denoting number of rows and number of columns in grid respectively (**1 ≤ R, C ≤ 20**). Then there will be **R** lines of **C** integers. Each integer will range from **0** to **5** (meaning of each of the number is given in the problem statement).

Output

For each case, output the case number followed by the answer to the query in the problem description.

Sample Input

```
3
3 4
0 0 3 0
3 1 5 0
0 0 3 0
3 4
0 0 3 4
3 1 5 0
0 0 3 2
3 4
0 0 3 0
2 4 5 3
0 0 2 0
```

Output for Sample Input

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

Problemsetter: Md. Mahbubul Hasan, Special Thanks: Jane Alam Jan