International Collegiate Programming Contest
South German Winter Contest

January 31, 2009

Problem Overview:

Good luck and have fun!

Problem A
Buffet

Time Limit: 1 second(s)

Traditionally, a contest dinner or a buffet is part of a regional contest. Being this year's contest organizer you have decided to have a buffet. Your caterer proposes to create a special buffet with n platters to serve m morsels. As no two morsels are alike, you can imagine that it will take quite some time to prepare the buffet. Of course, m is large enough to feed the always hungry teams from all over south western Europe.

One rule makes the buffet a special one: Each contestant must choose exactly n morsels of food, one from each platter. This rule is rigorously enforced by so-called "food watchers" who check every plate leaving the buffet. In spite of that rule, your caterer wants the contestants to still have the maximum freedom for choosing their meals. He therefore wants to distribute the m morsels to the n platters in such a way that the **first** person that chooses a meal can choose from the maximum number of different meals, i.e. combinations of n morsels.

Input

The first line of the input contains the number of test cases. Each following line contains n and m separated by a space ($1 \leq n \leq m \leq 10000$).

Output

For each test case output how many morsels have to be placed on each platter to maximize the number of different meals from which the first person in the buffet line can choose. Sort the number of morsels for each platter from smallest to largest.

Sample Input

```
3
2 2
2 4
3 7
```

Sample Output

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

Problem B
Candy Tycoon
Time Limit: 1 second(s)

A candy company wants to enlarge and freshen up their product line. The latest idea they have been working on is a completely new concept to arrange diverse candies in a quadratic box.
The candies that should be included into this amazing product have n different shells and n different fillings, where every shell and every filling can be combined to a total of $n \cdot n$ different candies.
For ages, candy box layouts have been designed to contain the candies in a symmetric alignment, but this is not what the management wants this time. Instead, the revolutionary idea is to find a layout so that in every row and every column each type of shells and each type of fillings appears exactly once.
The employees have been puzzling for a while to find the layouts for boxes of different sizes. But unfortunately, this is not so easy for mathematically untrained people. Therefore, the management decided to hire a computer scientist (you guessed it, it's you!) to solve that problem for $n \times n$ box sizes, where n is an odd number. The latter condition comes from the marketing department, which is convinced that customers won't ever buy boxes that contain an even number of candies.

Input

The first line of the input is a single integer c ($1 < c < 20$) that gives the number of test cases. The next c lines each contain a single integer n ($1 < n \leq 51$, n odd) that denotes the side length of the box (n also denotes the number of different shells and fillings for this test case, of course).

Output

If there is no valid layout for a given n, simply print "`no layout possible`". Otherwise, print an $n \times n$ matrix with a valid layout. Encode the fields of the box as follows: For the shells use upper case letters A, B, and so on. If n is larger than 26, continue after Z with lower case letters from a to z. For the fillings attach integers starting from 1 to n. As shown in the sample output, print a single space-separated valid layout.
Note that the output of two consecutive test cases should be separated by a blank line. If you find multiple solutions, any of these can be presented.

Sample Input

```
2
3
5
```

Sample Output

```
A1 B2 C3
B3 C1 A2
C2 A3 B1

C5 E2 B4 A3 D1
B1 D3 A5 E4 C2
E3 B5 D2 C1 A4
A2 C4 E1 D5 B3
D4 A1 C3 B2 E5
```

Problem C
Chef de Mensa

Time Limit: 6 second(s)

Being the mensa's chef is not easy. He has to decide which meals to cook so that all students are satisfied (which is nearly impossible as we all know). Even worse, he has to look at finances even if the mensa does not have to make much profit. Every day, he needs to choose from several dishes and has to decide how many of each meal to cook (we assume that each cooked meal is also sold). Moreover, each meal has different costs and some meals need to be made in the same devices which have limited capacity. There are even more constraints that add even more complexity to the chef's tasks. Since you are a computer science student, the chef has asked for your help in deciding which meals to cook.

Input

The first line holds two integer numbers $0 \leq n \leq 5$ and $0 \leq m \leq 10$. n is the number of different meals to choose from and m gives the number of constraints. The following n lines contain the net profit of each meal i in full Cents. Then m lines follow, each containing a constraint in the form of $k_1 * c_1 + k_2 * c_2 + ... < k_0$. The constraint can either be an equality (`=`) or an in-equality(`<, >`). The c_i (not given in the input) is the number of meal i that are cooked, k_i is an arbitrary integer constant. For example, the capacity of the oven may be that two times the number of meal *1* plus the number of meal *3* must not be greater than 100. This is written as `2 0 1 < 100`. For reasons of simplicity, we assume that each meal is cooked at most 40 times.

Output

In the first line, output the maximum profit in Cents that the chef can achieve. Then in the subsequent n lines print the c_i, i.e. how often each meal i has to be cooked to maximize the profit, sorted by their index i. All the meal numbers must of course be non-negative integers (there is no need to use floating point numbers). If it is not possible to cook any meal e.g. because not all constraints can be fulfilled, output `Impossible`. You can assume that the solutions are unique.

Sample Input

```
3 2
10
9
12
2 2 0 < 10
0 3 2 < 15
```

Sample Output

```
124
4
0
7
```

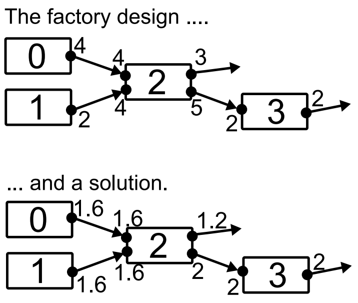
Problem D
Food Production

Time Limit: 1 second(s)

Food production these days is an extremely optimized process, usually performed by chemical technicians that use modern equipment which is similar to machines found in industrial automation. The fabrication of something edible is split up into several small steps, in which multiple ingredients are mixed and in which a chemical reaction produces one or more new ingredients, which are then fed into the next machine. To avoid waste of resources, usually not only one kind of food is produced, but a set of complementing ones, so that the side products of one kind of food are reused in creating another kind. As these food assembly lines can grow quite complex, your help is needed to calculate their throughputs.

You are provided with a description of the factory, which is built from multiple *food processors*. Each of them has pipes through which the chemicals (ingredients) are input and output. The inputs and outputs of the various processors are connected to build the food factory. The inputs and outputs of each processor are assigned numbers that indicate the amount of chemicals consumed or produced per hour. These indicate both the maximal values as well as the proportions. For processor 2 in the example figure this means that the chemicals provided at the inputs are consumed at the same rate. If there are 4 units of both input chemicals, 3 and 5 units of the output chemicals are generated. The machines can also be set to a lower production rate, which scales all input and output amounts by the same factor. To work seamlessly, the production rates of all machines have to be adjusted such that all connected inputs and outputs are dealing with the same amount of chemicals.

Your task is to find scaling factors for each processor, such that the overall output rate (the sum of all unconnected outputs) is maximized.



Input

The input consists only of one test case. The first line contains the number N ($1 \leq N \leq 10000$) of food processors. Each of the next N lines holds the description of one food processor. Each of these lines starts with two numbers I and O ($0 \leq I, O \leq 10$), giving the number of inputs and outputs, followed by $3I + O$ numbers $i_1, \ldots, i_I$, $o_1, \ldots, o_O$, $p_1, j_1, \ldots, p_I, j_I$. The i_k and o_k (all between 1 and 100 inclusive) give the per-hour-amount of input-ingredients and output-chemicals. The p_k and j_k describe for each input, where the ingredients are taken from (p_k-th food processor and its j_k-th resulting chemical, both zero-based).

You may assume that

- each resulting chemical is used as input by at most one other food processor,
- there are no loops and cycles (regardless of the production direction) in the assembly line, and that
- the input is consistent, *i.e.*, there are no indexing errors.

Output

Print the amount of all chemicals (food) produced per hour when the food factory runs at maximal capacity, rounded to exactly 3 digits. These are the values of unconnected output pipes.

(Sample Input and Output are provided on the next page.)

Sample Input

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

Sample Output

```
3.200
```

Problem E
Master of Cooking

Time Limit: 2 second(s)

Cooking is an art and few people can master the enormous challenge of scheduling all the preparations of the various parts of the meal (e.g. begin to stir, finish stirring, leave the house for shopping, return with goods, begin to boil, . . .). Thus, the nature of cooking requires to have a feeling of durations and time intervals. You are cooking novice and often lose track of time. Therefore, you wonder whether your meal will be ready on time before your guests arrive. You remember the time when you started cooking as well as the duration between many steps in the preparation of the meal. Of course you do not remember each and every duration between any two steps. So the question is when is the earliest and latest possible point in time when the meal will be finished?

Input

The first line of input contains the number of test cases. Each test case starts with a line that holds the number of total steps $2 \leq n \leq 10^4$ that are necessary to prepare the meal and the number of durations $0 \leq m \leq 10^5$ that you can remember. The next line states the time t_0 when you started to cook in seconds elapsed since 0:00:00 on January 1, 1970 (UTC) – of course cooking started before 11:00:00 on January 31, 2009 (CET). Each of the next m lines that follow contains three numbers a, b and c. a identifies an arbitrary step preceding step b. The steps are numbered consecutively from 0 to $n - 1$ but in non-chronological order. c denotes the time interval in seconds between steps a and b. Note that steps themselves are points in time. You may safely assume that meal preparations never take longer than 80 years.

Output

For each test case, print the earliest and latest point in time when the meal is finished on one line separated by a space. Times are denoted as seconds elapsed since 0:00:00 on January 1, 1970 (UTC). If you cannot determine an upper bound for the end of the cooking process then print "`never`" instead.

Sample Input

```
3
5 4
0
0 1 2
2 1 1
1 3 1
1 4 2
4 2
0
0 1 1
2 3 1
5 4
1
0 1 1
1 2 1
3 2 100
1 4 100
```

Sample Output

```
4 4
1 never
200 200
```

Problem F
Maximal Pizza Pleasure

Time Limit: 3 second(s)

Gwen loves Pizza. That is why she has moved into a flat located in the same house as her favorite pizzeria. Their pizzas are so exceptionally tasty that there is simply no business opportunity for any other pizzerias closeby. Although this sounds perfect at first, the downside is that you cannot get any pizza whenever the pizzeria is on holiday. After having suffered during holidays too often, Gwen decides to move again. She wants to have at least two pizzerias in her neighbourhood. Therefore, she rates all pizzerias of the city with scores from 1 to 9 (9 is best). She only considers pizzerias with a score of 5 or better. And she invents a way to measure the pizza pleasure of a prospective flat at place X and a pair of pizzerias A and B: The pizza pleasure is $p(X, A, B) = \frac{score(A)+score(B)}{dist(X,A)+dist(X,B)}$, where $dist(A, B)$ is the euclidean distance between A and B. The maximal pleasure of a point X and all pairs of different A and B is called "Pizza Value". Of course, Gwen needs your help in finding a new flat location X, where this pizza value is maximal.

Input

The input starts with the number of test cases n ($n \leq 10$). Each test case consists of one line with the number of pizzerias m ($m \leq 2,000$) followed by m lines. Each of these lines describes a pizzeria with its integer coordinates x, y, and its score s ($-10,000 \leq x, y \leq 10,000$).

Output

For each test case, output one line. This line consists of two integer values, the indices of the two pizzerias A, B that yield the maximal Pizza Value.
If there are multiple solutions possible, output the one with the smallest indices (lexicographical order).

Sample Input

```
4
3
0 0 9
4 8 9
9 2 5
3
0 0 9
4 8 5
9 2 9
3
0 0 5
4 8 9
9 2 9
3
0 0 9
4 8 8
9 2 8
```

Sample Output

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

Problem G

SDS, the Somewhat Different Sushi restaurant

Time Limit: 1 second(s)

Last year, a new sushi restaurant called *SDS, Somewhat Different Sushi*, has opened. As you might know, some people tend to make competitions out of everything. So SDS does. Each saturday, they arrange a two-person eating contest: winner is the person that eats the most **costly** sushi.
SDS has a special contest table. On this table, sushi plates are arranged in a row. The two competitors alternate in choosing a sushi plate to eat next. But they are only allowed to choose the leftmost or the rightmost plate of the sushi in the row. Both know the price tags of the individual sushi plates and both try to eat plates that in total sum up to the maximal price. SDS only charges the difference between the total values eaten by each contestant. The loser has to pay the bill.
My good friend Gerhard is well known for being a gourmet on the one side and eating as much food as possible on the other side. I will compete against him in this eating contest. Please help me to compute which order of plate selection is optimal for winning this contest. Of course, Gerhard – as a gourmet – selects the optimal sushi, as well. Knowing this fact, he is so generous to let me start the competition. If at some point during the contest it does not matter, we both choose the leftmost plate from the sushi row. There is never more sushi on the table than we both can eat.

Input

The first line holds the number of test cases T $(0 < T < 20)$. Each test case is given in two lines. The first line specifies the number of sushi plates S $(1 < S < 1000)$ lined up on the contest table. For a row of sushi plates (left to right), the second line gives the price P_i of each sushi plate in Cent $(0 < P_i < 2000)$, separated by single spaces.

Output

For each testcase, output two lines. First, print the amount that SDS charges for this eating contest (positive, if I won — negative, if Gerhard won the contest). Tell me in the second line, from which end of the row the plates are choosen (`L` for the left side, `R` for the right side).

Sample Input

```
5
2
10 20
2
20 10
3
50 10 10
5
3 40 2 4 8
10
99 11 88 77 44 55 66 100 33 22
```

Sample Output

```
10
RL
10
LL
50
LLL
-31
LLRRL
89
LRLLLLLLLL
```

Problem H

The Last Meal before a regional contest

Time Limit: 1 second(s)

Every year, several groups of programmers face a common problem: after having arrived at the city that hosts the regional ICPC-contest they need to find a place to eat. This is not as easy as it sounds as there are usually several restaurants in the vicinity but none is the obvious best option. The fact that different members of a group have different preferences makes it even worse. Some want vegeterian food, some want it cheap, some want it to be typical for the country they are in. . . Finding the option with the fewest disadvantages is not very easy. So they want you to write a program to do it for them. Your program also has to be flexible enough to allow for adding and removing of ranking criteria.

Input

The input starts with a line that holds the number t ($1 \leq t \leq 200$), the number of test cases. Each test case starts with a line containing n, m and k ($2 \leq n \leq 10$, $2 \leq m \leq 20$, $1 \leq k \leq 5$). n is the number of members of the group, m is the number of restaurants among which you have to find the optimum, and k is the number of features (e.g. availability of vegetarian food, price, or proximity) that are considered. The next n lines describe the preferences of each of the n group members by giving k values per line. These values can be 0, if this feature is not important at all to this member, 1 if it is of normal importance to this member, and 2 if it is very important and thus counted twice. Each test case ends with m lines describing the restaurants. Each such line again contains k values, followed by the name of the restaurant separated by a space. The values range from 1 to 5 with 5 being the best rating, 1 being the worst rating. Restaurant names consist only of small letters (a-z). They contain at most 30 characters and names are unique per test case.

Output

For each test case, output the names of the restaurants sorted by decreasing overall rating as follows: For each group member, the rating of the restaurant is the sum of all the ratings of the restaurant (counting the very important ones twice and leaving out the unimportant ones). For fairness reasons, this sum is divided by the sum of the member's preferences (and if it is all zero, it stays zero). The restaurant's total rating is the sum of all group member ratings. Print the names of the restaurants sorted by this overall rating. If two restaurants have the same overall rating, sort them alphabetically. Print an empty line between the outputs of different test cases.
(Sample Input and Output are provided on the next page.)

Sample Input

```
3
2 2 1
1
2
2 restauranta
5 restaurantb
3 3 3
1 2 0
0 1 2
2 0 1
5 5 5 thebest
4 3 4 aboveaveragefood
2 5 4 programmersrestaurant
2 3 2
1 2
1 0
3 5 bvdrfgxcdzwutpggkvnpqgtbkwyqmn
5 1 yqhiltiihztmnoeqilmitlhgeygghw
4 3 nbmxjkscrfkjfcfvndfldkwfmnvyqc
```

Sample Output

```
restaurantb
restauranta

thebest
aboveaveragefood
programmersrestaurant

bvdrfgxcdzwutpggkvnpqgtbkwyqmn
nbmxjkscrfkjfcfvndfldkwfmnvyqc
yqhiltiihztmnoeqilmitlhgeygghw
```

Problem I
The maximal munch

Time Limit: 1 second(s)

Gerard is a student that really enjoys eating. Gerard enjoys it so much that he often drives his fellow students nuts because he needs so much time in the cafeteria while they already want to leave for the lecture hall.
Gerard is not really picky about what he eats. But instead, the order in which he eats certain types of food matters a lot. For example, Gerard likes to begin with starters, followed by the main dish, and at the end the dessert. Therefore, for each day Gerard has a list that specifies in what order he can eat certain types of food. He does not necessarily eat all of them. But if he does eat n types of food, these have to be the first n items from the list, in exactly the order specified there. If he finishes pauses eating, he has to start over with the first item on the list. It does not matter whether he was through the list or not.

On weekends, Gerard usually visits some of his friends that really enjoy cooking. They will cook many varieties of food that he can eat if he wants to. His friends have other guests as well. So Gerard can skip some dishes if he needs a break or if that type of food does not suit his pattern. But Gerard has to restart on his list if he skips a dish. Otherwise he can eat as much food as he wants to as long as it fits to his list. Each time he finishes eating and begins with the first item on his list, Gerard needs a short break of at least one meal before he can start over. He needs this break so the food eaten can settle, therefore not for the very first time he starts eating.
Gerard always tries to eat as much food at those weekend parties as possible without breaking any of his rules. He asks you to write a program that gives the maximum number of dishes he can eat, given his list and the chefs' menu sequence.

Input

The first line contains the number of test cases that follow (at most 100).
Each test case starts with a line with two numbers g and c ($0 < g < 100; 0 < c < 10,000$). This line will be followed by a line with g integers separated by spaces. These numbers specify the order in which Gerard will eat food today. The final line of a testcase holds c integers separated by spaces. These numbers specify the order in which food is cooked today.

Output

For each test case, output one line with the maximum number of food types Gerard can eat without breaking any of his rules.

Sample Input

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

Sample Output

```
2
0
4
```

Problem J
Wait to be seated

Time Limit: 2 second(s)

Whenever a new resident arrives at the “Apartment Complex of Mayhem” (ACM), all the people living there celebrate this arrival with a dinner at Sushi Deli. Sushi Deli is famous for the best sushi in the city. If you ever come to San Diego be sure to try it!
Since many people know about that fact, there is always a long line in front of the place. To handle the masses of people, there is a list where each group can put one name and the number of people in the party. The restaurant has tables in different sizes. The groups will be seated according to the following rules:

- If a table gets free, the first waiting group that fits exactly to this table will be seated. If there is no such group, we take the first waiting group which is one person smaller than the table and so on.
- There is one exception to the rule above: We do not want a party to wait much more than 30 minutes. Thus, if a group waited already more than 30 minutes they will be seated to the next free table that is big enough. If two or more groups waited for more than 30 minutes, the one fitting best to the table will be seated. If two parties have the same size the free table will be assigned to the first on the list.
- If you have several tables available to seat a group, the party will be seated to the smallest yet fitting table.
- The seating of a group takes no time. Hence a group could be immediately seated in the moment of list entry.

After you have arrived at the Sushi Deli and have put your name on the list you want to find out your estimated waiting time. Is it enough time for a coffee in the Cafe on the other side of the street? From former visits you know that a party of two people usually stays for 20 minutes, a party of three for 25 and all bigger parties for 30 minutes in the restaurant after being seated. If I give you the waiting list of all entries from the opening of the restaurant this evening as well as a list of the tables, can you tell me after how much time I will be seated?
Note: Group and table sizes are 2 to 8 people. There will always be a table big enough for the biggest party given in the input. Your party is the last on the given list. The restaurant closes at midnight.

Input

The first line contains the number of test cases (up to ten) that are separated by blank line. Each test case starts with the number of tables t ($t < 100$) followed by t values that specify the number of seats per table. Then follows the number of parties as well as one line per party in the following form: Time of entry in the list (hhmm format in 24 hour notation), number of people in a party and one name without spaces. There are no entries at the same time. The list is sorted in chronological order.

Output

Output the estimated waiting time in minutes for being seated. If this would be at or after midnight, output “`No sushi today.`” instead.
(Sample Input and Output are provided on the next page.)

Sample Input

```
4
1 4
5
1900 3 Steve
1920 3 Chris
1930 3 Luis
1950 3 Daniel
2010 4 Awesomealmostnowaiting

1 4
5
1900 3 Steve
1920 3 Chris
1930 3 Luis
1940 3 Daniel
2010 4 WouldbeperfectbutDanielwaitedmorethan30min

1 6
4
0100 6 A
0110 2 B
0120 6 C
0130 6 D

2 2 4
3
2340 4 A
2350 2 B
2351 3 C
```

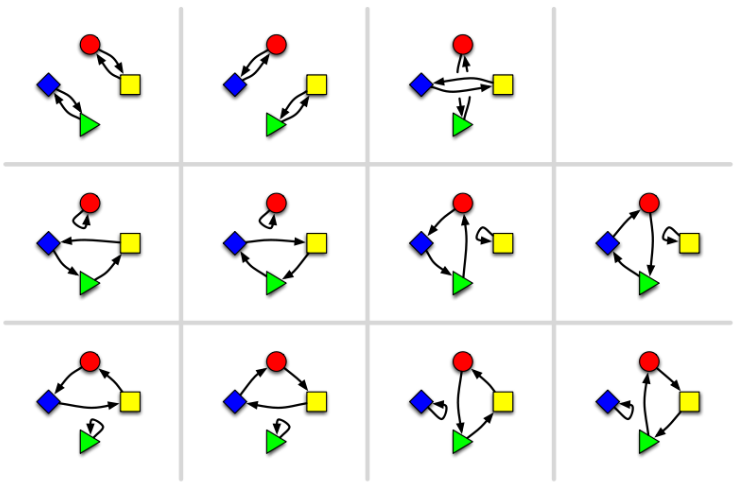
Sample Output

```
5
30
50
No sushi today.
```

Problem K
Waiters' Dance

Time Limit: 3 second(s)

Anna Hamilton is a wedding planer. She loves planing, coordinating, and directing these meaningful events. But her favourite part of the job is to organize the festive dinner in the evening. Lavish decorations fill the room where delicious food is served by well–trained waiters. Anna usually spends a lot of time thinking about the waiters and the tables they are serving. Actually she spents very much, too much time thinking about this problem. She usually tries to lay out all possible combinations of waiters and tables.
So let's assume she has to organize a small wedding where only four tables are necessary. The tables are internally labelled the *blue rhomb table*, the *red round table*, the *green triangular table*, and the *yellow quadratic table*. The parents of the bride can only afford two waiters. Mrs. Hamilton now draws the eleven possibilities of how two undistinguishable waiters can serve the four coloured tables. The result looks like this:



There are quite a number of constraints: Each waiter is assigned a certain number of tables, at least one. A waiter serves the assigned tables in a cycle. The starting point of the cycle does not matter. But the direction the waiter takes between the tables is important. This can be seen in the above figure. Whenever one of the two waiters has only one assigned table, there are two possible cycles in which the other waiter can work on the remaining three tables.
Drawing all possible combinations and cycles might be easy for four tables and two waiters but what happens with bigger weddings? Your task is to calculate the number of combinations for a given number of waiters and tables.

Input

First the number of test cases is given. Each test case is given in a single line and consists of the number of tables and the number of waiters. There are at least one table and one waiter. The number of waiters never exceeds the number of tables. At most 100 tables are served.

Output

Output the number of possibilities of how the waiters can serve the tables. Use a new line for each test case.
(Sample Input and Output are provided on the next page.)

Sample Input

```
8
1 1
2 1
2 2
3 1
3 2
3 3
4 2
9 3
```

Sample Output

```
1
1
1
2
3
1
11
118124
```