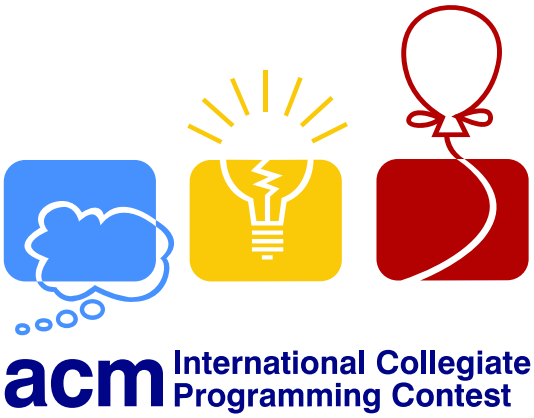
FAU Winter Contest 2012

February 04, 2012



The Problem Set

Good luck and have fun!

hosted by

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

sponsored by

accenture
High performance. Delivered.

This page is intentionally left (almost) blank.

Problem A
Beer Distribution

You have been given a new position as Assorted Construction Manager at the International Construction & Production Corporation. One of your new jobs is to ensure an appropriate beer supply at the newest construction site.

Construction will take place over several days; each day there will be a number of construction workers at the site. Each worker will require at least one bottle of beer. Furthermore, due to union rules, all construction workers have to be given the same number of bottles. As your boss might also be thirsty, you have to ensure that after distribution of the bottles to the workers, a certain number of bottles is left over.

Sadly, buying bottles must be handled by the internal corporate procurement committee. In order to minimize their workload, they require the number of bottles used daily to be the same for every day, even though that might lead to buying more bottles.

Given the number of days, the number of workers and leftover bottles for each day, you must determine the smallest number of bottles that fulfill these requirements, if this is possible at all.

Input

Input starts with the number of test cases on a line. Each test case starts with the number of days d ($1 \leq d \leq 250$), on a line. Then, d lines follow, each containing w_d (the number of workers on day d, with $2 < w_d < 2^{60}$) and l_d (the number of leftover bottles for day d, with $0 \leq l_d < w_d$). The least common multiple of the w_d values will be smaller than 2^{60}

Output

For each test case, print a line with the number of bottles to be used every day, if a solution exists. Otherwise, print a line with the string "`Impossible`".

Sample Input	Sample Output
2 3 3 2 4 2 5 1 2 4 3 6 2	26 Impossible

This page is intentionally left (almost) blank.

Problem B
Brick-Laying

More and more electrical and electronic engineering (EEE) students are feeling uncomfortable to study in the well-known but rather run-down EEE twin towers while the university is building a shiny, brand-new building for the maths department right in front of them. Thus, the university decided - money is since the introduction of tuition fees no issue - to also rebuild the twin towers.
The administration also considered students' desire for more colour on the campus. Instead of building another two concrete blocks, the administration will build wonderful reddish towers made of hundreds of thousands bricks.

The actual brick-laying will be carried out by professional brick-layers. For this step the university received offers from C different brick-laying companies. Each offer contains a list with the description of N_c brick-layers working at an individual company. Each brick-layer is characterized by his individual speed s, given as integral number in bricks per hour. Since the administration wants to build the towers simultaneously they asked the companies to split their workforce in two groups, so that each group of brick-layers is building one of the towers. Obviously the brick-laying companies all agreed on this desired approach, since it is how they would have done it anyway. With their vast experience that each company gained over decades they know how to split their team into two groups so that the added up speed of workers within each group is as equal as possible. The required time to build a tower is given by the accumulated group speed and the number of bricks. If a company has no more than one single worker this company will build the towers sequentially.
The problem for the administration is however, that they don't know how the companies will split up their workforces and thus cannot decide which company will be the fastest to build the two towers. The only thing they know is that for safety and social reasons brick-layers always create groups that do not differ by more than one worker. Every worker will be assigned to a group. If one group finished its tower faster than the other group they also won't help the other group to finish the second tower.
In order to choose the best brick-laying company the administration asked the EEE students for help.
After studying all their formularies the students agreed that there is no equation to solve this problem and they would need help as well. Obviously they do not want to go to the new maths department and ask for help there. So they are asking you to compute the minimal amount of time required by each brick-laying company to build the two towers, given the mentioned constraints.

Input

The first line denotes the number of test cases $1 \leq T \leq 10$. Each test case starts with a line giving the number of companies $1 \leq C \leq 10$ and bricks per tower $10^5 \leq B \leq 10^8$. A more detailed description of each company follows in the next C lines. Each company description starts with the number $1 \leq N_c \leq 50$ of brick-layers in company c followed by N_c integral numbers describing the speed $1 \leq s \leq 180$ of each brick-layer.

Output

For each test case output C lines with the minimal time in hours, accurate up to 10^{-4}, required by each company. Output a blank line between each test case.

Sample Input	Sample Output
2	7692.307692
2 1000000	9090.909091
4 50 60 70 80	
3 120 30 80	100000.000000
1 2000000	
6 10 9 9 6 4 2	

This page is intentionally left (almost) blank.

Problem C

Building Sky Scrapers

The *International Construction Project Company* (ICPC) builds really high sky scrapers, almost the highest in the world. As these sky scrapers are very vulnerable to severe storms or earth quakes, they hired a lot of stress analysts who design safe construction plans. Each construction plan gives the instruction when to use an ordinary brick and when to use some special material. However, the average construction worker cannot read or interpret the construction given by the stress analysts.
For each type of special material, they specify three characteristic numbers L, U, and I. A specific material is first used at height L, then at height $L + I$, then at height $L + 2 \cdot I$, ... But it is not used over height U.
Given the construction plan, your job is to transform it to a simpler version as follows: the workers ask you at which special material *type* is used at which height; they ask for the X-th occurrence of a special material, when counting all special materials (not types!) from bottom to top (the lowest occurrence is asked with 1). If more than one type of special material is used at one height, their "occurrence" will be counted in the same order as in the input. Due to the financial crisis, ICPC suffers from funding difficulties and will not use more than 100 000 special materials per sky scraper (even if the building is not safe then).

Input

The first line denotes the number of test cases $1 \leq T \leq 20$. Each test case starts with one line containing two integers S and Q ($0 < S \leq 10\,000$; $0 < Q \leq 1\,000$), where S gives the number of types of special materials used for this sky scraper. The next S lines specify the special material description by three integers L, U, and I ($0 \leq L \leq U \leq 10^{15}$; $0 < I < 10^{15}$) as specified above, each in one line. Then follows one line containing Q integers, each specifying a question by a construction worker as an integer Q_i ($0 < Q_i \leq 100\,000$).

Output

For each test case, answer the construction worker's questions, each in a separate line. If asked for special material X, first print the height where a special material (regardless of its type) is used for the X-th time. Then print its its type (indexed in the order as given in the input, starting from 1). You may safely assume, that we use at least X special materials (not types!) in the original construction plan.

Sample Input

```
3
3 5
0 20 5
8 15 3
12 16 2
2 3 7 8 11
1 3
0 999999999999997 99999999999999
1 7 2
2 3
0 999999999999999 123456789
5 999999999999999 987654321
50000 99999 75008
```

Sample Output

```
5 1
8 2
14 2
14 3
20 1
0 1
599999999999994 1
99999999999999 1
5486790073527 1
10973703603843 1
8231111111219 2
```

This page is intentionally left (almost) blank.

Problem D
Communication Chain

The renovation of the mensa (canteen) in Erlangen was not finished within the deadline and thus is not a shining example for other companies. One reason for the delay probably is bad communication. To avoid this problem in future projects, we need a guarantee that all the construction workers get the information they need. There will be one worker, namely the foreman, where all the information is accumulated. However, the problem is, that the foreman is not willing to inform every single worker about the current status and tasks. Therefore, all workers build something similar to a communication chain.

For example, there are four workers A, B, C, and D. The foreman is worker A. Then a *good* communication chain is: A distributes information to B, B distributes information to C, B distributes information to D. This communication chain is *good* because every worker gets all the information (sometimes indirect, see C and D). Note: the *chain* does not have to be exactly a chain while B distributes the information to two other workers. A *good* communication chain is defined as a communication chain that eventually distributes the information of the foreman to every worker.

The project manager is the same person, who controlled the renovation of the mensa. Please help him to answer whether his projects have *good* communication chains.

Input

The first line of the input gives the number of projects C ($0 \leq C \leq 100$). The first line of each project description holds the integers N and M: the number of construction workers in the current site ($0 < N \leq 1\,000$) and the number of information distributions ($0 \leq M \leq 100\,000$). Every worker has an unique index in the range $[0, N-1]$. Each of the following M lines holds two integers a_i and b_i that describe one information distribution. This means that the worker with index a_i distributes information to the worker with index b_i ($0 \leq a_i, b_i < N$, $a_i \neq b_i$). No information distribution will appear more than once. The foreman is the craftsman with index 0. There will be a blank line after each project.

Output

For each test case, print one line containing the string "`good`" in case the communication chain is *good* and the string "`bad`" otherwise.

Sample Input	Sample Output
4 4 3 0 1 1 2 2 3 4 5 0 3 3 1 1 2 2 3 2 1 5 5 0 3 3 1 1 2 2 3 2 1 3 2 1 0 2 1	good good bad bad

This page is intentionally left (almost) blank.

Problem E
Corn fields

The citizens of Roh, a small city in Arasia, recently discovered a number of magical artefacts while rebuilding their temple. They decided to use the artefacts in order to improve the fertility of their corn fields and asked a group of travelling mages for assistance. Every magician has his own kind of spell and may only use certain types of artefacts. Each spell requires one artefact in order to be cast. Moreover, every magician is only willing to cast at most a certain number of spells. The fertility of any field can benefit from at most a certain number of spells of the same kind, depending on its size. Additionally, one has to be careful not to cast too many spells onto a single field, as they might influence each other and produce unforeseen effects. Your task is to help the citizens of Roh to maximize the fertility of their fields by maximizing the number of spells cast.

Input

The first line of input contains a number T ($1 \leq T \leq 20$), the total number of test cases. The first line of each test case contains three integers n, m, f ($1 \leq n, m, f \leq 100$), where n denotes the number of types of artefacts, m denotes the number of mages and f denotes the number of fields. The second line of each test case contains n integers a_i ($1 \leq a_i \leq 1\,000\,000$), denoting the number of available artefacts of the different types. The next m lines describe the mages starting with two integers s and a ($1 \leq s \leq 1\,000\,000$; $1 \leq a \leq 20$), describe the number of spells the mage is willing to cast at most and the number of types of artefacts he can use, respectively. Then follow a additional integers n_i ($1 \leq n_i \leq n; n_i \neq n_j$), giving the ID of the artefact types that can be used.
Finally, f lines follow, each providing a description of a single field: The first two integers c and r in a line denote the maximum number of spells that can be cast on that field and the number of mages who can improve the fertility of the field, respectively ($1 \leq c \leq 1\,000\,000$; $1 \leq r \leq 20$). The rest of the line consists of r pairs of integers m_i and s_i, providing a description of how the fertility of the field can be improved by the mages. m_i denotes the ID of a mage who can improve the fertility of the field ($1 \leq m_i \leq m; m_i \neq m_j$) while s_i denotes the maximum number of spells of this mage which can improve the fertility of the field ($1 \leq s_i \leq 1\,000\,000$).
Adjacent test cases are separated by a blank line.

Output

Your program should output a single line per test case. The line should start with "`Case X: `" where `X` is replaced by the number of the test case. Then print a single integer representing the number of spells that can be cast.

Sample Input

```
2
1 1 1
2
1 1 1
2 1 1 2

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

Sample Output

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

The first case of the sample is trivial.
In the second case, mage 1 uses an artefact of type 1 to improve the fertility of field 1, mage 2 uses an artefact of type 2 to cast a spell on field 2, mage 3 uses an artefact of type 3 to cast a spell on field 3. The solution cannot exceed 3 although the total number of spells the mages are willing to cast is 4 in total; however, the first mage, who is willing to cast up to 2 spells, can only improve the fertility of field 1, which is possible at most once.

This page is intentionally left (almost) blank.

Problem F

Elevator Construction

You might know the "*Blaues Hochhaus*" (blue skyscraper) where most of the computer scientists of the FAU are working. There are three elevators in the building and most of the time at least one of the elevators is out of order and has to be maintained. A lesson learned from the construction of the new canteen, which was not finished in almost two years and still is delayed, the computer scientists do not risk the construction of completely new elevators.

After some research, the scientists discovered that the software is the main problem and far too complex. Thus, they decided to simplify the software part as follows: if a group of n persons arrives at the three elevators (in floor 0), every person specifies his/her target floor. Then, the group is split in four (possibly empty) subgroups, one for each elevator, and one that does not take the elevator at all. Every elevator goes either up or down (if possible) and stops in exactly one floor. Afterwards the elevators return to the default position (floor 0).
If the elevator does not stop at your floor, you have to walk up/or down the difference. Your job is to compute which elevator should stop at which floor while minimizing the total walked floors.

Input

The first line denotes the number of test cases $1 \leq t \leq 100$. Each test case consists of exactly two lines. The first line of each test case contains the number of persons n in a group ($1 \leq n \leq 50$). The second line contains n integers, where the ith integer specifies the target floor of the ith person. The target floor is an integer between -1 and 12 inclusively.

Output

For each test case, print one line containing four integers. The first integer gives the total walked floors for all persons in that group. The other three integers specify the floor number where the elevator stops (from left to right). The first two elevators can stop at floors 0 to 12 excluding floor 3. The third elevator can stop at floors -1 to 12. If there are multiple solutions that minimize the total walked floors, any will be accepted.

Sample Input

```
3
5
3 3 7 10 12
4
-1 3 3 12
4
-1 3 7 12
```

Sample Output

```
2 12 7 3
1 8 12 3
1 7 12 3
```

This page is intentionally left (almost) blank.

Problem G
IKEA

Building an IKEA kitchen is not as trivial as you might think. The construction manual is incomprehensibly written and some screws are always missing. However, the biggest problem is that the packages are always in the wrong place. The carrier placed all packages as one big stack in the corner of your kitchen. Unfortunately, you want to start the construction of your new kitchen in exactly that corner. Thus, you need to move the stack away.
However, the packages are very heavy. In one step, you can only move the topmost package from one stack to another stack. Furthermore, the carrier sorted the packages from heavy (bottom) to lightweight (top). You also do not want to place a heavier on a lighter package. There is enough space in the kitchen to create two more stacks.

Input

The first line denotes the number of test cases $1 \leq t \leq 100$. Each of the following t lines contains one integer p ($1 \leq p \leq 50$), the number of packages in the stack.

Output

For each test case, print one line containing the number of moves required to move every package from the original stack in the corner to a new stack (probably not in that corner) using at most one "helper" stack.

Sample Input	Sample Output
2 3 42	7 4398046511103

This page is intentionally left (almost) blank.

Problem H

Roofing Ceremony Cake

After almost 15 years, the *Absolute Construction Masters* Ltd. (*ACM*) finished your family home far enough, thus you are able to celebrate the roofing ceremony. You decided to invite both your complete family and your friends. For the dessert, you ordered a cake that has the shape of an equilateral triangle with side length a. You want to divide the cake into smaller equilateral triangle pieces which are served to the guests. The pieces need not be of the same size, but every guest (including you) receives exactly one piece and the complete cake must be used.
Furthermore, you know that there are exactly $5 \cdot (2^m)^2$ family members (you are one of these members, too) and exactly $1 + 3n$ friends on the ceremony. As you are too lazy to cut this cake by yourself into appropriate pieces, your wife will do this for you. However, you want to ensure that your family gets exactly $\frac{5}{9}$ of the complete cake in total. Hence, you divide the cake into two pieces beforehand with one cut. After this first cut, your wife must still be able to divide the two resulting pieces into the desired amount of smaller equilateral triangle pieces. Finally, you would like to know the length of this first cut.

Input

The first line denotes the number of test cases $1 \leq t \leq 2000$. Each of the following t lines contains the three integers a, n, m ($1 \leq a \leq 1337$, $0 \leq m \leq 32$, $0 \leq n \leq 10^8$).

Output

For each test case, print one line containing the length of your first cut. This number should be accurate up to 10^{-4} relative or absolute precision. If several suitable cuts exist, print the length of the longest one.

Sample Input	Sample Output
1 10 1 1	6.666666667

This page is intentionally left (almost) blank.

Problem I

Scratched Part Numbers

Bob stores all his building material in a huge pile in his warehouse. The materials are stacked in a chaotic way. But that is no problem, as every part has its part number printed anywhere on its package. It is quite inefficient to search for particular parts, but Bob always finds what he needs. Earlier today Bob had to find some steel rods. After searching them for hours, he graped them in a hurry, because he was already too late. Some of the rods got stuck and after pulling harder, he scratched parts of other packages part numbers with the edges of the rods. After delivering the rods, Bob returned to his warehouse and tried to fix the broken numbers. But he failed and now asks you for help. Beside the remains of the numbers, he shows you some catalogs, which contains all possible part numbers.

Input

In the first line the number n of part numbers in the catalogs and the number m of partial reconstructed part numbers is given ($0 < n, m \leq 100\,000$). n lines with one part number of the catalogs and m lines with one partial part number of one package follows. The part numbers of the different catalogs consist of up to 20 characters (only upper case letters, lower case letters, numbers and "`-`" are used). The part numbers in the catalogs are all unique. The partial part numbers of the packages consist of the same characters, but at some positions of the number the characters may be missing. The missing characters are given as spaces (one for every missing character). Luckily every part number has only one location of consecutive spaces.

Output

For every partial part number, print the complete part number, if you find a unique solution. Print "`not found`", if you cannot find a matching part number in the catalogs. If more than one catalogs part number could match the package part number print "`not unique`".

Sample Input

```
4 4
STUFF-1012-001
THING-0002-003
STUFF-2012-001
T112d1
THING-    -003
ST     012-001
STEEL 1012-001
     1
```

Sample Output

```
THING-0002-003
not unique
not found
T112d1
```

This page is intentionally left (almost) blank.

Problem J

Sewage Planning

Tim, a local engineer, has to plan the new sewage connection between two remote villages. As the two villages are located in the Great Plains, they are connected by a straight, flat street. Tim wisely decided to lay the sewage pipeline in parallel to the street to minimize the total length. To cover the whole distance, he got several quotes from tube producers including prices and lengths. Due to the extreme distance that must be covered, each tube variant is quite lengthy. Therefore, each distinct tube can only be delivered once. Now Tim has to select some of the quoted tubes such that the total price is minimal. The total length of all ordered tubes can match the total covered length, but it does not have to. On rare occasions, the total length of all tubes might not suffice to cover the whole distance and Tim admits defeat.

Input

The first line denotes the number of test cases t ($1 \leq t \leq 20$). Each test case starts with a line containing two integers d and n ($0 < d \leq 2\,000$; $0 < n \leq 2\,000$), where d denotes the total distance to be covered and n the number of tube models. Each of the following n lines contains a distinct tube t_i represented by its length l_i and its price p_i (both integers) ($0 < l_i \leq 999$; $0 < p_i \leq 100\,000$).

Output

For each test case, print one line. If the tubes cannot cover the whole distance at all, print "`IMPOSSIBLE`". Otherwise, first print the total cost of the solution, followed by a sequence of 0s and 1s where a 1 denotes that the corresponding tube (in the order of the input) is used for that solution while a 0 denotes that it is not used. If there are multiple solutions that minimize the total cost, any will be accepted.

Sample Input

```
3
50 4
10 20
5 5
30 90
5 10
20 4
5 5
10 10
15 20
5 10
999 2
555 1
443 1
```

Sample Output

```
125 1111
25 1010
IMPOSSIBLE
```

This page is intentionally left (almost) blank.

Problem K

Tunnel Maintenance

You are faced with the task of maintaining a long tunnel. Because you don't want to enter the tunnel yourself, you will use a robot to do the work. If the tunnel has length L, the robot is to enter the tunnel at one end (position 0) and leave the tunnel at the other end (position L), moving along the whole length of the tunnel in between to check for damages, do cleanup work, etc.

At each time, the robot is on a certain energy level (a nonnegative number). Its maximum possible energy level is M and moving/doing maintenance work along each unit length of tunnel causes its energy level to drop by exactly one unit of energy. If the robot's energy level ever drops to 0, it can't do any more work until its energy level is filled up again to a positive number.

Before entering the tunnel, the robot's energy level can easily be initialized to M outside the tunnel, at negligible cost. The length of the tunnel L might be greater than M, though, so the robot might get stuck inside the tunnel without further refilling. Fortunately, scattered along the tunnel, there are stations where the robot's energy level may be refilled automatically. Each such station has a fixed position between 0 and L inside the tunnel as well as a refilling cost per unit of energy. You may choose to fill up the robot's energy level to M at a station, partially fill it or completely ignore the station, but you can only refill by whole units of energy. You are interested in the optimum refilling strategy to minimize the total cost of maintaining the tunnel.

Input

The first line contains the number of test cases T ($T \leq 20$). The description of each test case starts with a line containing three integers L, M and N, denoting the length of the tunnel, the robot's maximum energy level and the number of stations along the tunnel, respectively ($1 \leq L \leq 10^9$, $1 \leq M \leq 10^9$, $1 \leq N \leq 10^5$). Following are N more lines, each one containing two integers p_i and c_i, where p_i is the position of station i and c_i is the refilling cost per unit of energy at station i ($0 < p_i < L$, $1 \leq c_i \leq 10^9$). The stations' positions are guaranteed to be pairwise distinct. Caution: There is a lot of input, don't use slow I/O methods!

Output

For each test case, print one line containing a single integer, the minimum total cost for the robot to maintain the whole tunnel or -1 if the tunnel can't be maintained by the robot because it would get stuck inside the tunnel for sure.

Sample Input	Sample Output
2 10 6 1 5 3 10 3 2 6 1 3 2	12 -1

This page is intentionally left (almost) blank.