International Collegiate Programming Contest
Lost in Space

20th of June, 2009

Problem Overview:

Good luck and have fun!

Problem A
Divisibility

Time Limit: 1 second(s)

In the year 2058, humankind finally meets intelligent beings from outer space. Well, with respect to maths they are not that smart. While every child on earth learns in primary school how to do basic divisibility testing against 2, 3, 5, and 10, they do not know such things. Their relationship to maths and numbers is so hopeless that they do not even have a common base when writing numbers. Of course, you want to help them. As you might know, there exists a divisibility test by calculating the (sometimes weighted) sum of digits for every number. The idea behind the test is, that you calculate the (weighted) sum of digits and then test that sum for divisibility. This can be recurred until one reaches zero (divisible) or any other number smaller than the divisor (not divisible). You will help the aliens by supplying a program that calculates the weights that are needed for deciding divisibility of positive numbers given the divisor and the base.

Input

The first line holds the number of test cases ($0 < t < 10$). Each test case consists of a line that holds two numbers, separated by a single space. The first number is the base b in which the alien calculates ($3 \leq b \leq 36$). The second is the divisor d the alien wants to be able to test divisibility against ($2 \leq d \leq 1000$).
You want to enforce them to calculate with base 10 – as most of the ordinary persons on earth do – so your program only has to be able to deal with numbers with base 10.

Output

For each test case, output one line containing the weights (the smallest non-negative number feasible) to be used when adding the digits, starting with the weight for the least significant digit, then for the second least significant digit, and so on. Separate weights by a single space. As numbers can potentially have an unlimited count of digits, the sequence of weights has to be infinite, too. However, it will be repetitive, thus you have to output the shortest possible sequence of weights that is to be repeated for the divisibility test. There might be some numbers before the period starts, so output a "p" to mark the start of the period.

Sample Input

```
3
10 3
3 3
10 7
```

Sample Output

```
p 1
1 p 0
p 1 3 2 6 4 5
```

Problem B

Hypergates

Time Limit: 2 second(s)

In the year 2058, hypergates became very popular to travel through space faster than light. For a connection between two planets exactly two hypergates are needed, one per planet. As hypergates are expensive and travelling through a sequence of hypergates is still unbelievable fast, the *Interstellar Company of Planet Connections* (ICPC) already determined the necessary hypergate connections and avoided cycles in their travel networks that must not be connected necessarily.

In order to save more money, the ICPC management works together with a consulting company. Millions of space dollars later, the consulting company came up with the advice to fire some teams of technicians. Instead of having one team at each planet that has hypergates, their proposal is to employ technicians only on necessary planets. They show in a study that technicians are seldom necessary at one side of the hypergate connection if there is a team at the other side of this hypergate connection. How many teams can be fired if the ICPC management follows this advice?

Input

The first line of the input holds the number of test cases T ($0 < T < 11$). Each test case is described as follows: the first line contains the number of planets P and hypergate connections C ($0 < P, C < 1000$). Each of the following C lines contains one hypergate connection in form of S T meaning that this hypergate connection allows passengers the faster-than-light travel from planet P_S to planet P_T and vice versa ($0 \leq P_S, P_T \leq P - 1$). Test cases are separated by a blank line.

Output

For each test case, print one line containing the number of teams that can be fired following the strategy of the consulting company.

Sample Input

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

8 7
0 5
1 6
2 6
4 7
5 6
7 3
7 6

10 8
0 1
2 4
3 8
5 9
7 6
8 1
8 7
8 9
```

Sample Output

```
2
5
5
```

Problem C

Lost Cargo

Time Limit: 2 second(s)

You are an employee of the interstellar delivery service *Planet Express* and deliver packages to the company's customers. On a tedious flight, you began to smoke cigars and to drink. As a matter of fact you should not drink and drive: By accident you have opened the loading hatch of your spaceship. On the long-range radar you can see that the cargo is spread all over the space quadrant. So you should better start collecting the lost cargo.

This task turns out to be tricky because after you captured a piece of cargo once, this piece is in danger of floating out of the space ship again while you try to capture the next piece. To solve this problem, you carefully bump the space ship into some target cargo that has a piece of cargo nearby (in the horizontally or vertically[1] neighboring space sector). The target cargo then floats towards the neighboring piece, collides and continues to float as combined cargo until interstellar drag causes the newly combined cargo to come to rest in the next sector (in floating direction). You continue to bump into cargo until all is concentrated in a single sector. Then you pick it up all at once. Bumping into cargo to just move it one or more sectors (i.e. if there is no cargo in the neighboring sector in bumping direction) is not an option because the floating cargo needs to be slowed down by crashing into the neighboring cargo. However, crashing into more than one cargo is not possible either. Instead it must come to rest in an empty sector next to the collision. Additionally, you have a preferred pick-up location, where your spare beer barrel has dropped out of the cargo bay.

The radar provides you with data in an abstract format. The quadrant is divided into $n \times n$ sectors. Each sector can either contain cargo, can be empty space, or can be a black hole that will suck in cargo in its sector[2]. Obviously, we will never push cargo into sectors with black holes or even out of the quadrant. Each sector has an associated coordinate (x, y), where x and y are in the range $[0, n)$. The x coordinate grows from west to east, whereas the y coordinate grows from north to south.

Input

There are at most 10 test cases in the input. The number of test cases is given on the first line. Then the test cases follow one after another. Each one starts with the number of sectors n on a line. All quadrants are square and do not have more than 10 sectors in either direction. The next line contains two integer values x and y that correspond to the desired pick-up location of the collected cargo. Then n lines follow that describe the space quadrant. The first line corresponds to the sectors from west to east in the far north. The following lines describe the sectors towards south. Each line consists of n characters, each describing one sector. A `.` character represents empty space, a capital `C` character represents a sector containing cargo, and a `#` character represents a black hole. There will always be at least one and never be more than 16 sectors that contain cargo.

Output

For each test case, print a list of bump operations that enable you to pick up the collected cargo at the pick-up location. Each bump operation is in the format "*x y* `from` *direction*" on a line on its own. The (x, y) coordinate specifies the coordinates, where the collided cargo will come to rest and *direction* is one of `north`, `east`, `south`, `west` and specifies from which direction the cargo originates. If there are possibly several solutions, any one is acceptable. If no such sequence of bump operations exists, print "`Cargo lost in space`" instead of a bump operation list. Print a blank line after each test case.

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

[1] Scientists will discover within the next 1000 years that space is in fact two-dimensional.

[2] Note that space ships can cross sectors with black holes if necessary.

Sample Input

```
3
3
2 2
.C.
.C.
C..
3
2 2
..C
CC#
...
7
3 3
##...##
##.C.##
...C...
.CCCCC.
...C...
##.C.##
##...##
```

Sample Output

```
2
1 2 from north
2 2 from west

Cargo lost in space

8
3 0 from south
3 2 from south
3 3 from west
3 1 from south
3 2 from north
3 3 from east
3 4 from north
3 3 from south
```

Problem D

Lost in Space

Time Limit: 10 second(s)

In the year 2058, Earth will soon be uninhabitable after the irreversible effects of pollution. Professor John Robinson, lead scientist of the Jupiter Mission, will lead a ship to the habitable planet Alpha Prime to prepare it for colonization by building a hypergate in orbit. A crew for the ship must be acquired quite urgently. There are many applications for the mission and there are quite a lot of different positions on the space ship to be filled. Luckily, one crew member can usually serve different positions, but on a concrete flight an applicant is assigned to only one position. Mission control has now to decide if all positions can be filled with all the available applicants.

Input

The first line contains the number of test cases $0 < t < 5$ that will follow. Each test case consists of the number of applicants $0 < n \leq 4000$ and the number of positions $0 < p \leq 2000$ separated by white space in the first line. The following n lines contain white-space separated lists of integers that indicate which positions each applicant can serve. Applicants can serve at most 10 different positions.

Output

For each test case print `It's safe to fly` if all positions are filled, or `We need more applicants` if some positions cannot be filled.

Sample Input

```
2
4 3
1 2
1
3
2
4 3
1 3
2
2
2
```

Sample Output

```
It's safe to fly
We need more applicants
```

Problem E

Mines

Time Limit: 3 second(s)

You are the chief engineer of the well-known interstellar mining company "Dig It". In the solar system of Betelgeuse there is a large, undeveloped asteroid belt that can be mined for various precious metals. As the chief engineer of the mining company you are given the task to establish mines on the asteroids. You have to determine the most profitable asteroids and mine them. There is one restriction to keep in mind. The mines on the asteroids cannot be located too close to each other because the mining equipment radiates subspace waves that may make the asteroids instable.

Input

The first line of input $n, 1 \leq n \leq 20$, denotes the number of asteroid belts to be processed, followed by the description of n asteroid belts. An asteroid belt is denoted as a $4 * m$ grid of numbers whereby each number denotes the profit $p, 1 \leq p \leq 10000$, (in space dollars) a mine on the asteroid generates per year. The description of the belt is preceeded by the number $m, 1 \leq m \leq 10000$. Mines cannot be placed on horizontally or vertically adjacent asteroids. It is possible, though, to place mines on diagonally adjacent asteroids. Your task is to compute the largest possible profit per year for your mining company. For the sake of simplicity, be aware of the fact that the asteroid $a_{i,1}$ and the asteroid $a_{i,m}, 1 \leq i \leq 4$ are **not** horizontally adjacent. The same holds each pair of asteroids $a_{i,1}$ and $a_{i,4}, 1 \leq i \leq m$. They are not vertically adjacent.

Output

Output the largest possible profit (in space dollars) for each asteroid belt on a single line.

Sample Input

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

Sample Output

```
8
32
```

Problem F
Ravenous Bugblatter Beast of Traal

Time Limit: 5 second(s)

With its fertile ground, beautiful vegetation, and great beaches for windsurfing, the Traal is a very attractive planet to all brave settlers in the universe. This led to many early pioneers living and dying on Traal. Some even claim that Traal was the first planet next to Earth where a human child was born. However, Traal's citizens face a nasty threat to their lives for a long time now: The ravenous Bugblatter beasts. These creatures simply eat anything and they certainly do not stop from eating a guy like you. While ravenous Bugblatter beasts often make a very good meal of (rather than for) visiting tourists, the citizens of Traal nowadays know quite well how important it is to always take a towel with them (a tool which also has a vast number of additional uses if you're lost in space). To deal with a beast, they just wrap the towel around their heads. This creature is so mind-bogglingly stupid that it concludes that if a person cannot see it, then it cannot see the person.
Once this knowledge has been established, the expectancy of life on Traal almost doubled. Still, being eaten by a ravenous Bugblatter beast stays second top cause of death on the planet (the first one being closely related to windsurfing); it may simply happen that a beast appears from behind and eats you as soon as you see it but before you have your towel in place. Therefore, for any new settlement to be founded, it is mandatory to implement an effective early warning system. Citizens have receivers in their towels – a radio-activated alarm goes off as soon as a beast is spotted. To spot beasts, a special sensoring was developed similar to a radar system. An array of towers encloses the settlement. Each tower is equipped with sensors such that the array covers the whole border. Every tower can track a beast in a circle around it with an individual radius, as towers are placed on hills where possible for better area coverage.

Recently, prospective areas for new settlements were found on Traal, each of them located close to beautiful beaches that are well suited for windsurfing. Based on the local terrain, for every area the most cost-efficient arrangement of sensor towers was determined. These are based on several rules: The tower array forms a convex shape, every coverage circle intersects with exactly two neighboring ones, and no tower lies inside the coverage of another tower. Your task is now to help with the further planning: write a program that gives the size of the safe area for the settlers. Considered safe is both the area covered by any early warning sensor and the area enclosed within the covered area.

Input

The input consists of consecutive array descriptions. The first line contains the number of arrays a, $a \leq 1000$. For every array, a line with a single number n, $3 \leq n \leq 1000$, the number of towers in the array is given. In the following n lines, three values are given per line: the coordinates x, y and the coverage radius r of each tower.

Output

For each tower array, output a single line containing the useful area, rounded to two decimal places. Within this solution, an error of 0.02 is still acceptable.

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

Sample Input

```
2
3
0 0 0.6
0 1 0.6
0.75 0.5 0.6
4
0 0 0.51
0 1 0.51
1 0 0.51
1 1 0.51
```

Sample Output

```
3.00
3.45
```

Problem G

Rayy

Time Limit: 3 second(s)

We are in the year 2058. Poker is still popular and is played with the same deck that has been used at the beginning of the century. Ten years ago, simple poker slot machines were very popular. But they are not used anymore because players have later found out how to win against them. As your friend Will is going to build a time machine you want to make profit out of it and play against those machines.

This machines played a poker variant called Rayy. It is played with the usual deck of cards. There are four suits (**s**pades, **h**earts, **d**iamonts, **c**lubs) and 13 values (**A**ce, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **T**en, **J**ack, **Q**ueen, **K**ing). One card for every combination of suit and value. Below, we use the bold face characters as abbreviation.

One round of Rayy consists of two steps: one dealing and a betting step. After 7 rounds, there is a final showdown phase. In a dealing step, both the player and the computer get one card simultaneously. This card can be "open" (both players can see the card) in rounds 3-6 and "closed" (only the player or only the computer can see the card) in the other three rounds.

In a betting step, the computer has to decide if it gives up its hand and leave the current "winners money" for the player. Alternatively, the computer can increase the "winners money" and give the player an "offer" [1]. Afterwards, the player has to decide. If he wants to play for the new "winners money", he has to pay the new "offer". Otherwise, he gives up his hand. When either the computer or the player gives up, the game is ended prematurely.

In the showdown after round 7, the computer and the player select five of their cards. The *lowest* poker hand wins[2]. If the player has more than two cards of the same value in his five cards, he looses. Hands with no pairs are compared in the following way:

- Both hands are sorted, highest value first, the suits are ignored. The order of the values is the following: $A < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < T < J < Q < K$
- The hand's values are compared lexicographical, from highest to lowest.
- When the player holds a lower hand, he wins and receives the current "winners money". Otherwise the computer wins and keeps all money.

The computer is easy to beat because it uses a very simple strategy. The computer keeps offering bets, as long as it is possible, that its selection of five cards contains no cards with the same value and no card higher than 9.

The strategy to play against this computer is to always pay the offer because it is possible that the computer gives up his hand due to the restrictions explained above. Only in the final betting round you want to use math. Given the "offer", the "winners money", your seven cards, and the four open cards of the computer you should decide if paying the "offer" has a positive expected value or not.

Input

The input starts with the number of test cases (at most 500) in one line. Each test case describes the situation in the 7th round, just after the computer has made an "offer". A test case consists of three lines. The first line gives the "offer" and the "winners money" (both positive and smaller than 2^{31}). Then a line with your seven cards follows. The last line of a test case lists the four open cards of the computer. The cards appear in the order in which they have been dealt.

As described above, cards are given by two letter codes for value and suit. All cards a separated by at least one white space.

You can be sure that the situation is valid according to the computer's strategy.

Output

You have to output one line for every test case. If the expected value of paying the "offer" is positive, output "`PAY!`". If the expected value is negative, print "`GIVE UP!`". If the expected value is 0, print "`nevermind`".

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

[1] The way in which "winners money" and "offer" are formed is complicated and not relevant for your task.

[2] Straights and flushes are irrelevant in Rayy.

Sample Input

```
3
1 100
As Ah 2d 2c 3s 3h 4d
8s 9s Ts Qs
100 101
7s 6s 5s 4s 3s 2s As
8s 9s Ts Qs
9312 9474
Ts Tc As 2s 3s 4s 5s
Th Td Ah 2h
```

Sample Output

```
GIVE UP!
PAY!
nevermind
```

Problem H

Shopping with a privateer

Time Limit: 1 second(s)

Doing business in space can be harsh. Some kind of business needs a special kind of people. Your old friend Lev is such a guy. He sells various goods in places far off the galaxy. And not all goods he sells are bought by him. Actually, most of them are not. He captures them from merchant ships, when their shields are malfunctioning. To increase the chances of such a malfunction, he shoots his lasers at their ships. This (and the fact that there are a lots of bounty hunters looking for privateers like him) is the reason that his vessel is always armed with the latest weapon technology.
Similar to the way Lev acquires his trading goods, he likes to buy the best weapons for the lowest prices. As space combat can easily mess up your weaponry he regularly needs to visit the local ship dealer to buy new cannons. Being a good space pilot but a horrible calculator, he asks you to advice him which cannon to buy. Of course, he wants a cannon that gives the most damage for the lowest price, i.e. it is your job to find the one with the least damage-for-the-buck-ratio.

Input

The first line holds the number of test cases ($0 < t < 10$). Each test case consist of one set of weapons of which you should find the one best suited for the privateer to buy. The first line of each test case will have a single number n ($1 < n < 100$) on it denoting the number of weapons available in this test case. This line is followed by n lines, one for each weapon holding two numbers p ($0 < p < 10,000$) and d ($0 < d < 5,000$). p is the price of this weapon in credits and d the damage done by this weapon in a very obscure unit not further explained.

Output

For each test case, you should output the index of the weapon with the best ratio of damage to price on a single line. The first weapon on the list has index 1 and so on. So if the best weapon is the third of the input list, just output 3. If there is more than one weapon on the list with the same ratio, output the index of the first of them. Do not print any blank lines between test cases.

Sample Input

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

Sample Output

```
2
1
```

This page is intentionally left (almost) blank.

Problem I
Space Parks Holidays

Time Limit: 3 second(s)

> *Welcome to our Space Parks Holidays Information Center. Space Parks Holidays consists of hundreds of thousands holiday centers scattered all over the known universe and even more Space Cruisers completing your vacation experience...*

Something like that will be displayed at the start screen of our new Information Center, scheduled to be launched tomorrow. Our holiday entertainments (parks and cruisers) are grouped in three categories. The *cheap & funny* category for `low`-waged clients, the *adequate & comfortable* category for `average`-waged clients, and the *luxurious & exclusive* category for the `high`-waged society. Those categories are not fixed but can be separated by cruiser and park prices. The differences between each low-waged entertainment to each averaged-waged one are more than 100 space credits, as it is between the average-waged and the high-waged ones. Within each category no such gap can be detected due to our huge number of choices.
Unfortunately, our chief developer was a saboteur hired by our business rival "Interstellar Travelling Services". He deleted the core access routines to our database. Now we have just plain data about our space parks (location and cost) and our huge fleet of cruisers (connecting our parks), but our clients cannot access and extract the interesting statistical details. For example, to find out how many space parks (resp. space cruisers) we are offering that match the personal travel budget of our clients.
Please help us in answering such queries during the next 5 hours!

Input

Each input consists of one network description and many user queries. The first line holds the number of space parks P and space cruisers C ($0 < P, C < 100000$) separated by a single space. After this line, there follow P lines describing the price p in space credits ($0 < p < 100000$) for each park (starting from park with id 1). After the park descriptions C lines of cruiser descriptions follow. Each cruiser description consists of the ids of two parks that are connected by the cruiser and the price p in space credits, separated by a single space.
The remaining up to 100000 lines are queries. Each query starts with the type (`RANGE`, `PARK`, or `CRUISER`) followed by one or two parameters. `RANGE` expects one parameter out of `low`, `average`, or `high`. `PARK`/`CRUISER` expects a single price value or two price values representing a price range (in space credits). Input is ended by EOF.

Output

For each query, print the result in a single line. A `RANGE` query returns the corresponding price range. A `PARK` (resp. `CRUISER`) query returns the numbers of space parks (resp. cruisers) of each category that match the given price range. Output prices are immediately followed by ``sc'', ranges are separated by ``-'' and listings by ``, ''. If a query has no result, the string ``no result'' will be displayed. See sample output for more details.

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

Sample Input

```
6 8
800
400
530
160
361
120
1 2 120
1 4 480
3 4 400
3 6 520
5 6 260
1 5 400
6 4 160
1 3 800
RANGE low
CRUISER 120 400
RANGE average
PARK 260 361
RANGE high
CRUISER 40 400
PARK 800
PARK 420 500
```

Sample Output

```
120sc-260sc
3 low, 2 average
361sc-530sc
1 average
800sc
3 low, 2 average
1 high
no result
```

Problem J

Universal Languages

Time Limit: 20 second(s)

On one of your recent trips through the galaxies you have discovered an abnormally looking apparatus. This gadget has a small screen to display a message. Messages change every single time unit. A message consists of a lot of symbols but the scientists are unsure who made the apparatus and which language it uses.
In order to guess the right language, they ask you to help by searching the number of occurrences of known symbols, syllables and words from different languages.

Input

The first line holds the number of test cases ($0 < t < 11$). Each test case is started by a line holding the number of symbols S in the message, the symbol width W and the number of queries Q ($0 < S < 100000$, $0 < W < 100$, $0 < S \cdot W < 100000$, $0 < Q < 100$). A symbol therefore consists of W single characters (from `a` to `z`). Two symbols are considered equal if each single character is equal. After this line, there follow S symbols in one or more lines (not necessarily split on symbol boundaries). After this follow Q lines containing the query symbols forming the words.

Output

For each test case, print Q lines (one per word in the query). Print how often that word appears in the message followed by the word itself.

Sample Input

```
3
29 1 3
icpcprogrammingcontestsarefun
icpc
contest
cp
7 1 4
aaabaab
aab
a
b
aa
16 3 3
ccc
abacccccccc
abacccccccccc
abacccc
abaxxccccc
aba
ccc
xxc
```

Sample Output

```
1 icpc
1 contest
2 cp
2 aab
5 a
2 b
3 aa
3 aba
11 ccc
0 xxc
```