GCPC 2013

15.06.2013

The Problem Set

No	Title
A	Boggle
B	Booking
C	Chess
D	Kastenlauf
E	No Trees But Flowers
F	Peg Solitaire
G	Ringworld
H	The King of the North
I	Ticket Draw
J	Timing
K	Triangles

Good luck and have fun!

hosted by

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

sponsored by

accenture High performance. Delivered.

Problem A
Boggle

I am sure, you are a big fan of the board game "*Boggle*". Don't worry if you are not familiar with the rules, I will explain them to you. A Boggle is a 4×4 grid of letters where your job is to find as many words as you can. If I play Boggle with (or against) my wife, she always wins – the loser (that's me) then always has to do all these little jobs like to take out the trash. So, please help me to win again.
Words in a Boggle can be constructed from adjacent letters (i.e. horizontally, vertically and diagonally), but the same dice may only be used once in a word. The words have to be listed in our dictionary to be valid.
Words with 3 or 4 letters count 1 point, words with 5 letters 2 points, 6 letters 3 points, 7 letters 5 points. 8 letter words will give 11 points. If you find more than one word (and I hope you do), points will be summed up to form your score.

Input

There is only one test case per file. The first line contains the number of words w in the dictionary ($1 < w < 300\,000$). Then follow w lines, each containing one word. Words consist of up to 8 upper case letters ('A'-'Z'). After the dictionary specification, there is a blank line. Then follows the number of Boggle boards b in one line ($1 < b < 30$). Each boggle is given as a 4×4 grid of upper case letters in four lines. Boggles are separated by blank lines.

Output

For each boggle, print one line containing the maximal possible score, the longest word and number of found words. Words that are twice (or more often) in one Boggle, only count once. If there is more than one longest word, print the lexicographically smallest one. You may safely assume that there is at least one valid word in each Boggle.

Sample Input

```
5
ICPC
ACM
CONTEST
GCPC
PROGRAMM

3
ACMA
APCA
TOGI
NEST

PCMM
RXAI
ORCN
GPCG

ICPC
GCPC
ICPC
GCPC
```

Sample Output

```
8 CONTEST 4
14 PROGRAMM 4
2 GCPC 2
```

This page is intentionally left (almost) blank.

Problem B
Booking

Pierre is in great trouble today! He is responsible for managing the bookings for the ACM (Accomodation with Moderate Costs) hotel and has just realized that the booking software has a severe bug. It has created overlapping and wrong room assignments. Pierre is now worried that the hotel might be overbooked. Since the software manufacturer is neither very responsive nor competent, he must check this himself, so that he can timely take countermeasures if necessary.
Fortunately, Pierre was able to export all original bookings (including reservation codes plus correct and valid dates of arrival and departure). The only information that got lost is the time of the booking placements, so that Pierre cannot retrieve any booking priorities (e.g., first-come-first-serve). Using the available information, could you please help Pierre out and tell him a room assignment with the minimum number of rooms that satisfies all bookings? Please be aware that a room must always be cleaned before reuse. Since Pierre does not want to take any risks, he tells you to only consider the maximum cleaning time.

Input

The input consists of several lines. The first line contains $1 \leq T \leq 100$, the number of test cases. Each test case starts with a line containing two integers $1 \leq B \leq 5\,000$, the number of bookings, and $0 \leq C \leq 360$, the cleaning time (in minutes) for one room.
Then follow B lines, each containing the reservation code (a random alphanumeric string of up 20 characters) and the dates of arrival and departure of one booking. Dates are given as strings in the format "YYYY-MM-DD HH:MM" (see example test case), where reservations are only for the years 2013 until 2016.

Output

For each test case, print the minimum number of needed rooms on a single line without any additional blanks. Be aware of leap years but ignore daylight saving time.

Sample Input

```
4
2 120
1 2013-07-01 15:59 2013-07-08 16:30
2 2013-07-08 17:30 2013-07-15 12:00
3 60
65 2013-07-08 14:30 2013-07-08 16:00
32 2013-07-01 16:00 2013-07-15 12:00
91 2013-07-01 16:00 2013-07-08 15:00
2 360
a7 2016-02-21 14:00 2016-02-28 21:00
xx 2016-03-01 01:00 2016-03-02 12:57
2 60
a9 2016-02-21 14:00 2016-02-28 11:00
a8 2016-02-28 12:00 2016-03-11 21:00
```

Sample Output

```
2
3
1
1
```

This page is intentionally left (almost) blank.

Problem C
Chess

In chess the bishop is the chessman, which can only move diagonal. It is well known that bishops can reach only fields of one color but all of them in some number of moves (assuming no other figures are on the field). You are given two coordinates on a chess-field and should determine, if a bishop can reach the one field from the other and how. Coordinates in chess are given by a letter ('A' to 'H') and a number (1 to 8). The letter specifies the column, the number the row on the chessboard.

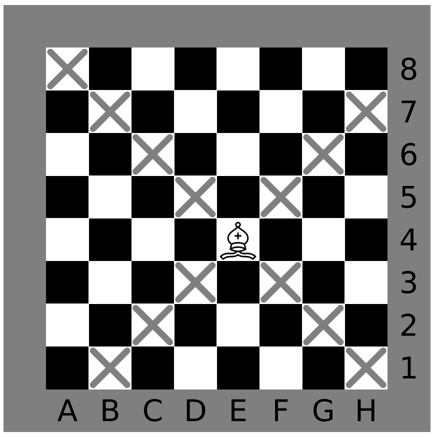

Figure 1 – Chessboard, bishop and fields the bishop can reach in one move

Input

The input starts with the number of test cases. Each test case consists of one line, containing the start position X and end position Y. Each position is given by two space separated characters. A letter for the column and a number for the row. There are no duplicate test cases in one input.

Output

Output one line for every test case. If it's not possible to move a bishop from X to Y in any number of moves output '`Impossible`'. Otherwise output one possible move sequence from X to Y. Output the number n of moves first (allowed to be 4 at most). Followed by $n + 1$ positions, which describe the path the bishop has to go. Every character is separated by one space. There are many possible solutions. Any with at most 4 moves will be accepted. Remember that in a chess move one chessman (the bishop in this case) has to change his position to be a valid move (i.e. two consecutive positions in the output must differ).

Sample Input

```
3
E 2 E 3
F 1 E 8
A 3 A 3
```

Sample Output

```
Impossible
2 F 1 B 5 E 8
0 A 3
```

This page is intentionally left (almost) blank.

Problem D
Kastenlauf

Once every year, Jo and his friends want to visit the local fair in Erlangen, called Bergkirchweih. This year, they want to make a Kastenlauf (box run). They start at Jo's home, and have one box (Kasten) of beer (with twenty bottles). As they are very thirsty, they drink one bottle of beer every 50 metres.
As the way from Jo's home to the Bergkirchweih is pretty long, they need more beer than they have initially. Fortunately, there are stores selling beer on the way. When they visit a store, they can drop their empty bottles and buy new bottles, but their total number of full bottles will not be more than twenty (because they are too lazy to carry more than one full box).
You are given the coordinates of the stores, of Jo's home and of the location of the Bergkirchweih. Write a program to determine whether Jo and his friends can happily reach the Bergkirchweih, or whether they will run out of beer on the way.

Input

Input starts with one line containing the number of test cases t ($t \leq 50$).
Each test case starts with one line, containing the number n of stores selling beer (with $0 \leq n \leq 100$). The next $n + 2$ lines cointain (in this order) the location of Jo's home, of the stores, and of the Bergkirchweih. The location is given with two integer coordinates x and y, (both in meters, $-32768 \leq x, y \leq 32767$).
As Erlangen is a rectangularly laid out city, the distance between two locations is the difference of the first coordinate plus the difference of the second coordinate (also called Manhattan-Metric).

Output

For each test case print one line, containing either "happy" (if Jo and his friends can happily reach the Bergkirchweih), or "sad" (if they will run out of beer on the way).

Sample Input

```
2
2
0 0
1000 0
1000 1000
2000 1000
2
0 0
1000 0
2000 1000
2000 2000
```

Sample Output

```
happy
sad
```

This page is intentionally left (almost) blank.

Problem E
No Trees But Flowers

Tom and Sarah soon have their second anniversary and he wants to buy her something special. Even though she's very keen on any kind of tree, he thinks that this probably is too big of a gift. So, he decides to buy her flowers. But not the lame cut off ones that will die after two weeks tops. No, he wants to buy her flowers that may last forever. Flowers in a flower pot. Now, he has a problem. Finding the right flowers for his girlfriend isn't that difficult. However, finding the right flower pot turns out to be much more complicated. Since he wants the flowers to live as long as possible, he needs to find the best-fitting flower pot he can find. When he comes to the Arboreal and Crops Market, he notices that for each bouquet of flowers there is an optimal size written for the flower pot. However, the offered flower pots do not state their volume. The only information the shop assistant can provide is the outline of each flower pot given as a function. Since Tom was never very good in math and his date is in less than an hour, he asks a nearby customer for help: You. Fortunately, you brought your laptop on your shopping tour. Can you help Tom?

The outline of a flower pot is described by a function $f(x) = a \cdot e^{-x^2} + b \cdot \sqrt{x}$, where x is the vertical distance from the bottom of the flower pot. The body of the flower pot thus is defined by the rotational body created when rotating the graph of f around the x-axis. The height of a flower pot is denoted by h.
The volumes of two flower pots differ at least by 10^{-4}. Also there is only one flower pot closer to the optimal than any other one, with an accuracy to 10^-4.

Input

The input always contains one test case which consists of several lines.
The first line contains a decimal number $0 < V \leq 10^5$, the optimal size for Tom's favorite flower, and an integer $0 < N \leq 5$, the number of flower pots available.
The next N lines each contain three decimal numbers describing the corresponding kth flower pot: $0 \leq a_k \leq 10$, $0 \leq b_k \leq 10$, and $1 \leq h_k \leq 10$.
For the sample input the volumes are approximately 34.72348 and 21.77966.

Output

Print one line containing a single integer that describes the index of the best fitting flower pot. The index of the first flower pot is 0.

Sample Input

```
25.0 2
1.0 2.0 2.0
2.0 1.0 2.0
```

Sample Output

```
1
```

This page is intentionally left (almost) blank.

Problem F
Peg Solitaire

The game of peg solitaire, popular at the court of the French king Louis XIV, has the following rules. Given a two-dimensional board with a mesh of holes, each hole can contain one peg (pin). The only legal move of a peg is a vertical or horizontal jump over an adjacent peg into the empty hole next to the jumped peg in line with it, which is then removed. The original goal of the game was to leave a single peg in the predefined position on the board by performing only legal moves. Obviously, such a solution is possible only for certain board forms and starting configurations. To drop this restriction, we slightly redefine the problem: Given the starting configuration of the board, determine the minimum number of pegs that can be achieved by means of legal moves as well as the minimum number of moves required to reach that number of pegs.

Input

The first line of the input contains one number, $1 \leq N \leq 100$ which represents the number of test cases. Each test case is described by the following lines of input that represent the initial state of the solitaire board.
In this representation '`.`' denotes an empty hole and '`o`' a hole with a peg in it. '`#`' indicates a part of the board without a hole. All boards have the same shape, see sample input (that includes position of holes). In its initial state, the board can contain at most 8 pegs. There is an empty line between two consecutive test cases.

Output

For each test case your program should output a line with two numbers separated by a single whitespace, with the first one denoting the minimum number of pegs achievable by legal moves starting with the given initial state, and the second one providing the minimum required number of moves.

Sample Input

```
3
###...###
..oo.....
.....oo..
.........
###...###

###...###
..oo.o...
...o.oo..
...oo....
###...###

###o..###
.o.oo....
o.o......
.o.o.....
###...###
```

Sample Output

```
1 3
1 7
1 7
```

This page is intentionally left (almost) blank.

Problem G
Ringworld

The world is actually neither a disc or a sphere. It is a ring! There are m cities there, conveniently called $0, 1, 2, \ldots, m-1$, and arranged on the ring in the natural order: first 0, then 1, then 2, ..., then $m-1$, and then again 0 (as the world is a ring, remember?). You are given a collection of contiguous ranges of cities. Each of them starts at some city x, and contains also cities $x+1$, $x+2$, ..., $y-1, y$, for some city y. Note that the range can wrap around, for instance if $m = 5$, then $[3, 4, 0]$ is a valid range, and so are $[1]$, $[2, 3, 4]$, or even $[3, 4, 0, 1, 2]$. Your task is to choose a single city inside each range so that no city is chosen twice for two different ranges.

Input

The input consists of several lines. The first line contains $1 \leq T \leq 20$, the number of test cases. Each test case consists of a number of lines. The first line contains two integers $1 \leq m \leq 10^9$ and $1 \leq n \leq 10^5$ denoting the number of cities and the number of requests, respectively. The next n lines define the ranges: the i-th row contains two integers $0 \leq x_i, y_i < m$ describing the i-th range $[x_i, x_i + 1 \mod m, \ldots, y_i]$.

Output

For each test case, output one line containing YES if it is possible to assign a unique city to each request, and NO otherwise.

Sample Input

```
4
3 3
0 1
1 2
2 0
200000 3
100000 100000
100001 100001
100000 100001
6 6
0 1
1 2
2 3
3 4
4 5
5 0
6 6
0 0
1 2
2 3
4 4
4 5
5 0
```

Sample Output

```
YES
NO
YES
NO
```

This page is intentionally left (almost) blank.

Problem H
The King of the North

Winter is coming (or going? who can be sure these days) and a new king rises in the North. The message travels quickly these days... That is why you, the rising king, have not much time left. You need to rally your bannermen behind you. But one question seems harder to answer than you would have first expected. How large of a kingdom can you claim and how many men should you send for? Your advisors have taken a close look at the potential kingdom and have determined how many of your bannermen would be required to fully defend any part of the map against your foes. As you are a loving and caring king, you want to minimize the number of men that have to serve in your army. To give your war council a fair chance of figuring out the best kingdom to defend, you have to determine the size of the army that you will raise as soon as possible.
Luckily, armies are not that advanced yet. You will only have to defend against armies moving horizontally or vertically (an army cannot pass but your bannermen diagonally). Your kingdom counts as defended, if there is not a single way to reach your castle, starting anywhere outside of the map, without passing to a fully defended area. Squares on the map labeled 0 represent high mountains, or walls, no one would ever be foolish enough to climb. You can assume to be save from invasion without sending any bannermen to defend them. Since you are uncertain about what lurks behind the wall (or in our case the borders of the map), you have to assume the worst and plans as if you would never be able to hold any position outside of the given map.

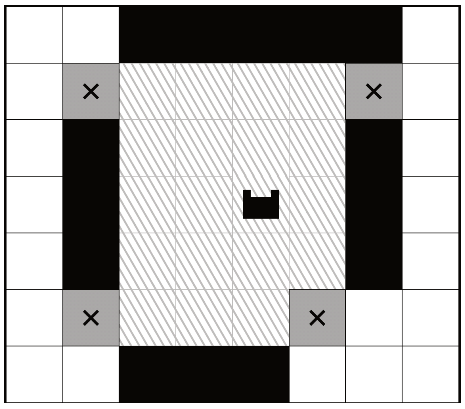

Figure 2 – Illustration of the sample input — the kingdom can be defended with a minimal army of 37 bannerman, located at the cross-marked positions. The kingdom itself is illustrated using a tiling pattern. Note that you do not have to find out about the kingdom, or the position of you bannermen. These questions are to be figured out by your war council.

Input

The input is given in the form of the (rectangular) strategic map which your advisors came up with. Every square in map is assigned a number of bannermen which would be required to defend the position against any potential army. The map is formatted as follows: In a first line you are given to integers R and C, $3 \leq R, C \leq 300$, specifying the dimensions of the map. This line is followed by R lines, each containing C integers $0 \leq c_i \leq 100\,000$, the number of bannermen necessary to defend each square. Finally, you are given $0 < r < R-1$ and $0 < c < C-1$, the position of your own castle on the map.

Output

Output an integer on a single line, the smallest possible army you would require to defend any kingdom.

Sample Input

```
7 8
42 42  0  0  0  0  0 16
42 11 14 42 42 42 10 16
42  0 42 42 42 42  0 16
42  0 42 42 42 42  0 42
42  0 42 42 42 42  0 42
42 11 42 42 42  5  5 42
42 42  0  0  0 42 42 42
3 4
```

Sample Output

```
37
```

Problem I
Ticket Draw

The concert promoters of the Bon Jovi Tour 2013 have decided to sell tickets for the concerts in lotteries. The rules are quite simple. For every concert, fans can apply online for tickets. In response they receive unique reservation numbers. It is important that for each concert the numbers distributed online are consecutive nonnegative integers starting with 0. Unfortunately, as the organizers tried to draw reservation numbers randomly, they discovered that the pseudo random generator they used is extremely slow. To minimize the number of calls to the generator, they invented a peculiar but fair enough way to distribute tickets.

As soon as the reservation for a concert is finished, the organizers ascertain the number of submissions M and draw one random integer Z from $\{0, \ldots, M-1\}$ (remember, fans get integers from 0 to $M-1$). Integer Z is the only object the organizers have to draw randomly! Finally, to complete the selection rules the organizers determine an integer $r > 0$ which has a direct impact on the number of selected tickets.

Now, using Z and r, tickets are selected deterministically as follows. For the reservation numbers $0, \ldots, M-1$ and the number Z, their decimal representations of length n are considered, where n is the length of the representation of $M-1$ without leading zeros. Thus, the decimal representations of the remaining numbers are padded on the left with leading zeros, if needed. If $z_1 \ldots z_n$ denotes such a representation for Z, then the holder of a number $a_1 \ldots a_n$ gets the ticket if and only if the strings $z_1 \ldots z_n$ and $a_1 \ldots a_n$ have a common contiguous substring of length r or more which starts at the same position. Speaking formally, he or she gets the ticket if there exists an index i, with $1 \leq i \leq n-r+1$, such that $z_i \ldots z_{i+r-1} = a_i \ldots a_{i+r-1}$. For example, if $Z = 56743$ and $r = 3$ then a fan with a reservation number 06740 gets a ticket, but a fan having 56143 does not.

Your task is to help the organizers to estimate, for given numbers M, Z and r, the exact number of tickets selected in such a way.

Input

The first line contains the number of concerts C, with $1 \leq C \leq 5000$. Then follow C lines, each containing three integers M, Z, and r, with $0 < M \leq 10^{18}$, $0 \leq Z \leq M-1$ and $r \geq 1$. You may safely assume that r is smaller or equal to the length of the decimal representation of $M-1$.

Output

For each concert, print one line containing the number of tickets selected during the ticket draw.

Sample Input

```
8
89 32 1
67 49 1
67 45 2
1000 23 1
1000 401 2
1000 54 2
3571 2 3
3571 976 3
```

Sample Output

```
18
15
1
271
19
19
13
12
```

This page is intentionally left (almost) blank.

Problem J
Timing

A clash of galaxies is coming!
A galaxy ruled by the mysterious MdI is trying to annex our milky way, but the galactical government has plans to turn things round.
Our intelligence agency has infiltrated the enemy's headquarter and gained surprising intelligence. The enemy is moving its units around according to a fixed scheme: for each fort a fraction of the units is moved to other forts in each time unit (the time of the flight is negligible).
Now the government has fixed a time when to attack. Your order is to compute the weak points. But as the enemy's galaxy is far, far away it takes one time unit to fly there. Furthermore, we are also certain that the MdI will recognize our target and immediately start all ships which can reach our attacking point (via one link, regardless of its direction). The spy informed you that these strengths are only statistical values, i.e. some sort of indicator as float.

Input

The first line of the input contains the number of test cases ($1 \leq T \leq 10$). Each test case starts with one line containing three integers stating the number of enemy forts N ($1 \leq N \leq 100$), the number of links l ($0 \leq l \leq (N-1)^2$) and the time from now when to attack t ($0 \leq t \leq 5000$). The second line contains N doubles u_i ($0 \leq u_i \leq 1000$) specifying the strength of the stationed troops at each fort followed by l lines containing the links. Each link is described by two integers s_j ($0 \leq s_j < N$), t_j ($0 \leq t_j < N$) describing the source and the target of the link and one double p_j ($0 < p_j \leq 1$), the fraction of units transferred from s_j to t_j in each time unit.

Output

Print the lowest indicator of the enemy galaxy with an absolute or relative error less than 10^{-6}.

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

Sample Input

```
3
4 3 1
100 200 10 305
0 1 0.25
1 2 0.1
2 0 0.75
4 3 1
100 200 10 312
0 1 0.25
1 2 0.1
2 0 0.75
4 4 5
100 200 300 400
0 1 0.2
1 2 0.2
2 3 0.3
3 0 0.2
```

Sample Output

```
305.000000000
310.000000000
659.879000000
```

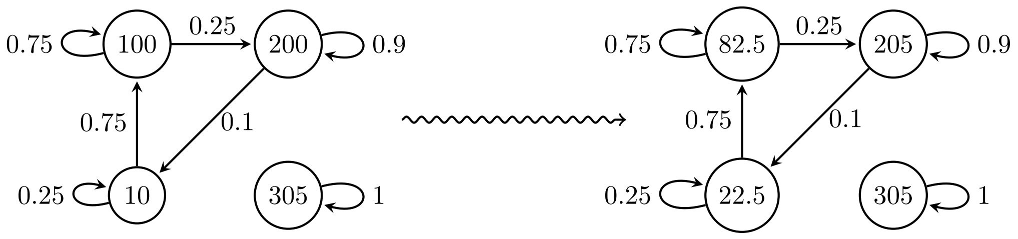

Figure 3 – The statistical strengths of the first sample before and after the first time step.

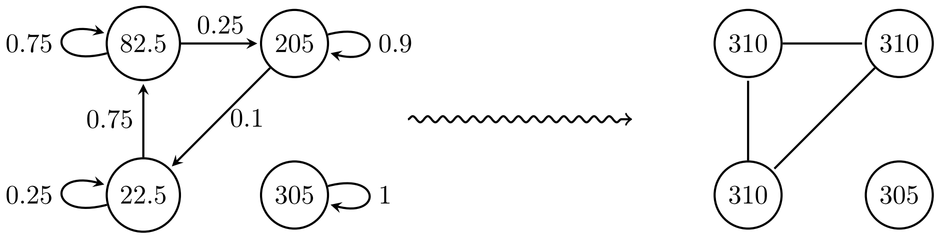

Figure 4 – Strength of the forces to face at each fort. Note that links are used in both directions.

Problem K
Triangles

You got a very strange gift for your birthday: two triangles in the three-dimensional space. Each triangle consists of three infinitely thin segments, and each segment stays straight no matter how hard you press it. Now, you actually wanted to get just one triangle, so you try to move the triangles far apart from each other, possibly after rotating one or both of them, so that you can throw away one of them. Is it possible? Or are they tangled?

Input

The input consists of several lines. The first line contains $1 \le T \le 1000$, the number of test cases. Each test case consists of two lines. The first line contains 9 integers $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3 \in [-1000, 1000]$ denoting the vertices of the first triangle. The second line contains another 9 integers $x'_1, y'_1, z'_1, x'_2, y'_2, z'_2, x'_3, y'_3, z'_3 \in [-1000, 1000]$ denoting the vertices of the second triangle. Both triangles will be non-degenerate, which means that the corresponding triples of points will not be colinear. Moreover, it is guaranteed that no pair of segments from two different triangles intersects, and there is no common plane containing both triangles at once.

Output

For each test case, output one line containing YES if the triangles are tangled, and NO if it is possible to move them very far apart from each other.

Sample Input

```
2
0 0 0 10 0 0 0 10 0
1 1 10 1 1 -10 10 10 0
0 0 0 10 0 0 0 10 0
11 0 0 0 11 0 11 11 1
```

Sample Output

```
YES
NO
```

This page is intentionally left (almost) blank.