Problem A
Ntonic Sequence

Source file: sequence.{c | cpp | java}
Input file: sequence.in

A sequence is monotonic if it is either strictly increasing or decreasing. For example, <2, 4, 8, 9> or <9, 6, 5, 3> are monotonic sequences. A bitonic sequence is a sequence that first monotonically increases, then monotonically decreases or vice-versa. For example, <4; 6; 8; 3; -2>, <9; 2; -4; -10; -5> are bitonic sequences. A sequence is polytonic, if it is neither monotonic, nor bitonic. For example, <1; 3; 12; 4; 2; 10> is a polytonic sequence. Given a sequence of *n* integers, your task is to determine whether it is mono, bi, or polytonic sequence.

Input

The input consists of several sequences of numbers. Each sequence starts with *n* (0< *n* <= 1000), the number of integers in the sequence, followed by *n* 32-bit signed integers separated by whitespace and/or newline. Input is terminated by a sequence having *n*=0, which should not be processed.

Output

For each sequence, you are to output one line, containing the sequence number, a space and only the type, i.e., “mono”, “bi” or “poly”.

Sample Input

```
4 2 4 8 9
6 1 3 12 4
   2 10
5 9
2 -4 -10 -5
0
```

Output for Sample Input

```
1 mono
2 poly
3 bi
```

Problem B
If It's Meant To Be, You Will Guess It

Source file: `guess.{c | cpp | java}`
Input file: `guess.in`

There was this girl everyone wanted to get her number, but she wouldn't give her number that easily. If she didn't want to give her number to someone, she would answer "If it's meant to be, you will guess it". In some cases, she would give only the first few digits of her number and say "If it's meant to be, you will guess the rest of it". In this problem, your task is to find out the chances someone can guess her number correctly.

Given the number of digits in her phone number N ($0 < N < 20$) and the number of known digits as described above X ($0 \le X \le N$). Knowing that each digit has an equal chance of being between 0 and 9 inclusive, and given Y ($0 \le Y \le 10^9$) number of unique trials to guess the phone number, what is the probability that you get the number right?

Input

The input starts with a number T ($0 < T < 1{,}000$) that represents the number of test cases in the file. Each test case consists of one line that contains three integers N, X, and Y, respectively.

Output

The output for each test case is in this form:

```
k. ans%
```

where ***k*** represents the test case number (starting at 1), and ***ans*** is the probability the phone number is guessed correctly in percentage printed with accuracy of 2 decimal points.

Sample Input

```
3
3 0 10
7 4 2
7 0 2000
```

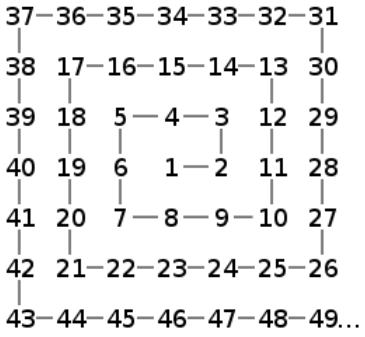
Output for Sample Input

```
1. 1.00%
2. 0.20%
3. 0.02%
```

Problem C
Ulam's Spiral

Source file: `spiral.{c | cpp | java}`
Input file: `spiral.in`

Stanislaw Ulam was the first to notice back in 1963, that placing the natural numbers in a spiral as shown below:



creates a layout where prime numbers tend to appear along diagonals.

The question is whether the same technique can be used to generate very large prime numbers from arbitrary locations on the spiral.

Your task is to list a sequence of numbers arranged in a diagonal, given the coordinates of the starting location on the spiral (1 is at the axes origin, i.e. location (0,0)), the angle of the diagonal (45, 135, 225 and 315 degrees) and the length of the sequence. For example, the following input:

0 0 45 4

should produce: 1 3 13 31

Input

The input file starts with line containing a number $N<100$ that represents the number of input cases. N lines follow, each containing four integers *x*, *y*, *angle* and *len*. Coordinates *x* and y are integers with a maximum absolute value of 10^9. The *angle* can be one of 45, 135, 225 or 315, and the *len* is a non-negative number less than 100.

Output

For each input case you should output *len* numbers in a separate line, starting with the one at position (*x*, *y*).

Sample Input	Output for Sample Input
3 1 1 45 3 0 0 135 10 2 2 225 5	3 13 31 1 5 17 37 65 101 145 197 257 325 13 3 1 7 21

Problem D
The Big Sorter

Source file: `sorter.{c | cpp | java}`
Input file: `sorter.in`

The Borrows-Wheeler transform is a technique used in compression. The first step of the transform requires the sorting all the possible rotations of the input data. If this were to be applied to the string "CONTEST", we would get the following rotations:

CONTEST

ONTESTC

NTESTCO

TESTCON

ESTCONT

STCONTE

TCONTES

Which when sorted would produce the sequence:

CONTEST

ESTCONT

NTESTCO

ONTESTC

STCONTE

TCONTES

TESTCON

with the original string at position 0.

Your task is to calculate the position of the original string in the sorted sequence of the rotations. If a rotation matches the original string, the original is considered smaller.

Input

The first line of the input file contains an integer N ($0 < N < 100$) indicating the number of test cases. Each test case consists of one line containing an arbitrary string of ASCII characters. The maximum line length is 64kB.

Output

For each test case you should output in a separate line, the position of the original string in the sorted sequence of all the possible rotations.

Sample Input

```
2
CONTEST
PROGRAMMING
```

Output for Sample Input

```
0
8
```

Problem E
GPA Scare

Source file: `scare.{c | cpp | java}`
Input file: `scare.in`

Abdulla C. Mo is in his senior year in the university, but some bad results in the previous semester have dropped his GPA. Risking losing his chance to join a graduate program, he tries to calculate the grades he needs for his last courses.

For example, if he has completed 100 credits so far, with a GPA of 2.9 and he needs a GPA of 3.0 to get into a graduate school, for the last four courses (each course counts for 3 credits) of his study, he needs:

A, A, A-, A-

given that A corresponds to 4, A- to 3.7, B+ to 3.3, B to 3, B- to 2.7, C+ to 2.4, C to 2 and C- to 1.7.

If multiple combinations of grades satisfy the required target GPA, the smallest possible ones should be listed, sorted in descending order. So a B, B combination is preferable to an A, C one. The lowest acceptable grade is C-.

Input

The first line of the input file contains an integer N ($0 < N < 10{,}000$) indicating the number of test cases. Each test case consists of one line containing four numbers : current GPA ($0 \le G \le 4$), target GPA ($0 \le G' \le 4$) , completed credits ($0 \le C \le 300$), and remaining courses M ($0 \le M \le 100$).

Output

For each input case you should output the sorted list of minimum grades that can achieve the target GPA in a separate line. If a solution cannot be found, you should output “Target GPA unattainable”.

Sample Input

```
3
2.9 3 100 4
3 3.3 100 8
3 2.4 60 10
```

Output for Sample Input

```
A A A- A-
Target GPA unattainable
C- C- C- C- C- C- C- C- C- C-
```

Problem F
Robots for Mines Destruction

Source file: `robots.{c | cpp | java}`
Input file: `robots.in`

Robots have come to replace humans in many physically demanding and dangerous tasks. Clearing minefields is one such endeavor where robots have proven to be invaluable. In this problem you are given to task of clearing a minefield, where a previous sweep has pin-pointed the locations of all the mines.

The minefield is modeled as a square grid of *n x n* dimensions. The grid positions are numbered in sequence starting from 0, and in a left to right and a bottom-up fashion. The location of each mine is represented by the corresponding grid number.

The problem is that the robots available can only move upward or right-ward, starting from grid position 0. This means that a single robot may not be enough to cover the whole minefield. You task is to calculate the minimum number of robots required for the task.

Two examples are shown below, with mines indicated by the gray cells:

Example 1: n=3, with m=3 mines at positions: 4, 5, 8

6	7	8
3	4	5
0	1	2

Solution:

1 robot is needed, the first path is: 0->4->5->8, note that we do not care how the robot reaches position 4.

Example 2: n=3, m=6 mines at positions 1, 2, 4, 5, 6, 8

6	7	8
3	4	5
0	1	2

Solution:

3 robots are needed, the first path is: 0 -> 1 -> 2 -> 5 -> 8, the second is 0 -> 4, and the third is 0 -> 6.

Input

The first line contains an integer *k* that represents the number of problem instances followed by a number *n* that represents the dimension of the grid (all problem instances assume the same *n*) . Each problem instance is given as a set of numbers; the first represents the number *m* of the mines, followed by *m* numbers that represent the position *p* of each mine on the grid.

$1 < n \leq 100$
$0 \leq m < n^2$
$0 \leq p < n^2$

Output

For each problem instance, your program should output one line in the format:

`Case k: ans`

where ***k*** represents the test case number (starting at 1), and ***ans*** is the minimum number of robots required to clear the field.

Sample Input

```
4 5
25
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
3
0 4 24
4
8 13 5 11
8
2 4 24 7 13 11 22 23
```

Output for Sample Input

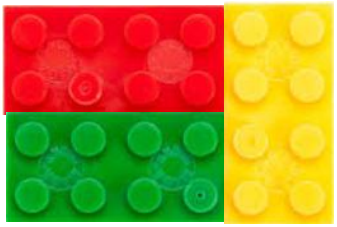
```
Case 1: 5
Case 2: 1
Case 3: 2
Case 4: 3
```

Problem G
Lego Structure

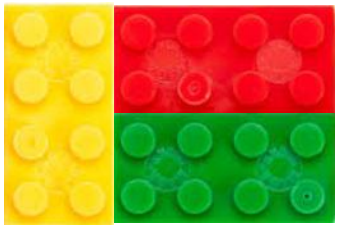
Source file: `lego.{c | cpp | java}`
Input file: `lego.in`

Lego structures are constructed using several Lego pieces stacked on top of each other. In this problem, you will be given two layers of Lego pieces, and you are asked to check if these two layers form one structure or not. Each Lego piece is of rectangular shape and has *a* rows of pins, each has *b* pins. Two Lego piece will form a structure (become stuck to each other) if one pin from top layer is position on top of another pin at the lower piece.

Example 1
The following two layers form one structure, the first layer is composed of three pieces, each of size 2x4 (or 4x2) pins. The position and orientation of every piece is fixed as given. Pieces in every layer are given in a way to form a rectangle with no empty spaces anywhere. If the two given layers below are stacked on top of each other, they will form one structure.



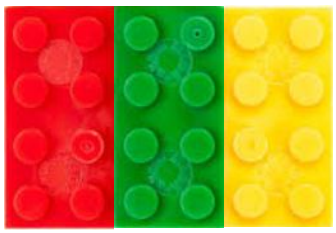
Bottom layer



Top layer

Example 2
The following two layers do not form one structure, since if they are stacked on top of each other, the rightmost piece of every layer will form a separate structure, i.e. it can be separated from the structure without removing pieces from each other.

Bottom layer



Top layer

In each problem, you will be given two rectangular layers of Lego pieces, the two layers are of the same size, and completely full of Lego Pieces (no spaces in between). You are not allowed to manipulate any piece in any layer. All Lego piece are of rectangular shape, and they can be of any size, all given layers are also of rectangular shape.

Input

The input file will consist of *k* problem instances. The first line contains an integer *k* that represents

the number of problem instance. Followed by a line that has two integers represents the number of pieces for top layer *n* and bottom layer *m*, followed by *n* + *m* lines each contain four integers: x_0, y_0, x_1, y_1 that represents the Cartesian coordinates of the location of every piece, where (x_0,y_0) represents lower left corner, and (x_1, y_1) represents upper right corner. All pieces are located on the positive side of the X-Y coordinate plane, and both lego layers should be positioned with the left bottom corner at the origin. Next instance starts with two positive integers: *n* and *m*, followed by another (*n* + *m*) lines, each contains the coordinates of the two corners of the piece. The instances are terminated after *k* problem instance. All Lego blocks are of rectangular shape. Lego pieces of one layer are given in no particular order.

$1 \leq n \leq 1000$
$1 \leq m \leq 1000$
$0 \leq x0, x1, y0, y1 \leq 10000$

Output

For each problem instance, your program should output a single line consisting of the letter V for valid structures and I for invalid ones, terminated with empty last line.

Sample Input

```
3
3 3
0 0 4 2
0 2 4 4
4 0 6 4
0 0 2 4
2 0 6 2
2 2 6 4
3 3
0 0 4 2
0 2 4 4
4 0 6 4
0 0 2 4
2 0 4 4
4 0 6 4
2 2
0 0 10 2
0 2 10 5
1 0 10 5
0 0 1 5
```

Output for sample input

```
V
I
V
```

Problem H
Gulf Soccer League

Source file: `gsl.{c | cpp | java}`
Input file: `gsl.in`

A number of teams competing for Gulf Soccer League in a round robin league style based on points, where every team play all other teams at home and away. A winner in every game is rewarded 3 points, loser 0 points, and 1 point for each team on draw. In this problem, you will be given a number of teams, the results of all rounds of play, and you are asked to provide the rank of teams at the end of the league.
In order to obtain the overall performance of teams, the following two rules should be applied: *team performance rule* and *team rank rule*:

Team performance rule:
The ranking of any two teams is determined by applying the following criteria in sequence, until a tie is broken:

1. The final number of points accumulated by each team. The team with higher points performed better.
2. The total outcome of their direct two games result: the team that scored more goals in the two matches performed better.
3. The team that scored more away goals in their direct games performed better.
4. The difference between goals scored and goals received in all games is used, the team with higher goal difference performed better (difference can be negative).
5. The total number of goals scored by the team, the team that scored more goals performed better.

If all tie breaker steps in ***Team performance rule*** fail, the two teams are said to have the same performance.

You are asked to rank teams based on their performance from 1 to 4, rank 1 for the best performance, and 4 for the worst, using the following method:

Team rank rule

> **A team with a given rank must have performed better than all teams with worst ranks using *Team performance rule above*, and all other teams with better rank must have performed better than this team.**

If two or more teams have the same performance, *or the Team rank rule cannot be used to rank them*, the teams share the same rank (i.e. 1), and the next rank is not assigned (i.e. 2) , and next team shall take the following rank (i.e. 3).

Example 1
Teams will be given as upper case letters A, B, C, and D, results will be given as integers for number of goals scored by each team in the format: A 3 B 1 for the result of A win over B is 3-1.

Input: the results of 12 games for 4 teams (A, B, C, D).

A 1 B 0
C 1 D 0
A 2 C 1
B 0 D 1
A 2 D 2
B 1 C 0
B 1 A 0
D 3 C 1
C 0 A 2
D 1 B 0
D 1 A 1
C 1 B 0

Results table

Rank	Team	W (win)	D (draw)	L (Lost)	P (Points)	GF Goals for	GA Goals against	GD Goal Diff
1	D	3	2	1	11	8	5	3
2	A	3	2	1	11	8	5	3
3	B	2	0	4	6	2	4	-2
4	C	2	0	4	6	4	8	-4

In this table, D performed better than B and C using ***Team performance rule 1.*** Also D performed better than A using ***Team performance rule 3*** based on their direct results of the two games (1 – 1 and 2 – 2), Rule 1 does not give a result since they have the same number of points, rule 2 does not give a result since the total result of their two games is (3-3), rule 3, gives a result since D scored 2 goals away (2-2), while A scored only 1 goal away (1-1).

Team A performed better than both B and C using rule 1. B performed better than C using rule 4 GD (-2 for B and -4 for C). Note that rules 1 to 3 cannot be used between B and C.
The teams should be ranked as follows

1. D.
2. A.
3. C.
4. B.

Example 2
The same results above, except that the two games between A and D ended with the same result, i.e. (A 2 D 2) and (D 2 A 2), then, Teams and B should have the same rank, and the teams should be ranked as:

1. A.
1. D.
3. C.
4. B.

If two or more teams share the same rank, they must be given that same rank number in the output, and printed in alphabetical order.

The following simplifications are assumed for all problem instances:

1. All teams are named with single upper case letters starting from A.
2. You are given only 4 teams, 12 games total per league.
3. Games are given based on round-by-round, where in every round every team plays one game (a total of 6 rounds, two games in every round).
4. Rounds are not given in any particular order.
5. You are given the complete set of games for all round for all teams (12 games total).
6. Highest number of goals that can be scored is 9, the lowest is 0.

Input

The input file will consist of *k* problem instances, followed by k sets of league results. Each contains the result of the 12 games between 4 teams in the following format: T1 N1 T2 N2 (letter followed by a number followed by a letter followed by a number). Problem instances are separated with an empty line.

'A' ≤ T1, T2 ≤ 'D'
0 ≤ N1, N2 ≤ 9

Output

For each problem instance, your program should output 4 lines shows the rank of each team followed by the team name followed by a dot. If two or more teams share the same rank, they should be printed with the same rank on separate lines in the alphabetical order. Print an empty space between problem instances. The last line should be an empty line.

Sample Input

```
2
A 1 B 0
C 1 D 0
A 2 C 1
B 0 D 1
A 2 D 2
B 1 C 0
B 1 A 0
D 3 C 1
C 0 A 2
D 1 B 0
D 1 A 1
C 1 B 0

A 1 B 0
C 1 D 0
A 2 C 1
B 0 D 1
A 2 D 2
B 1 C 0
B 1 A 0
D 3 C 1
C 0 A 2
D 1 B 0
D 2 A 2
C 1 B 0
```

Output for sample input

```
1. D
2. A
3. B
4. C

1. A
1. D
3. B
4. C
```

Problem I
Largest Number

Source file: `largest.{c | cpp | java}`
Input file: `largest.in`

You are asked to generate the largest integer number by arranging a given list of numbers. For example, given 10, 20, 3, 9, the largest formed number is 932010.

Input

The input file will start with a positive integer *k* represents the number of problem instances. Each problem instances starts with an integer *n* followed by *n* numbers. Each number, *m* is a nonnegative number less than or equal to 999999

$1 \leq k \leq 1000$
$1 \leq n \leq 100$
$0 \leq m \leq 999999$

Output

For each problem instance, your program should one line that contains the largest possible integer. The last line should be empty.

Sample Input

```
4
2 123 55
3 51 52 59
4 1 2 3 999
2 7 74
```

Output for sample input

```
55123
595251
999321
774
```

Problem J
Pedestrian Bridges

Source file: `bridges.{c | cpp | java}`
Input file: `bridges.in`

The city is planning to introduce pedestrian bridges instead of regular crosswalks. This will help achieve smoother traffic and increase the safety for pedestrians. The project will start with the busiest crosswalks to replace them with pedestrian bridges. The city needs your help to find out which crosswalks are the busiest.

You will be given details of how many pedestrians cross a specific crosswalk, the times of their arrival, and how long they take to cross. Your task is to calculate the maximum number of pedestrians on the crosswalk at any point of time. A pedestrian is considered on the crosswalk from the time they arrive until the time they leave the crosswalk inclusive.

Input

The input starts with a number T ($0 < T < 1{,}000$) that represents the number of test cases in the file. Each test case starts with a line containing an integer N ($0 < N \leq 100{,}000$) representing the number of pedestrians for a specific crosswalk. ***N*** lines follow each containing two integers X_i and T_i ($0 \leq X_i, T_i \leq 10^9$) representing the time of arrival, and the amount of time needed to cross for pedestrian ***i,*** respectively. Pedestrians' arrival times will be given in non-decreasing order.

Output

The output for each test case is in this form:

```
k. ans
```

where ***k*** represents the test case number (starting at 1), and ***ans*** is the maximum number of pedestrians on the crosswalk at any point of time.

Sample Input

```
2
4
1 20
2 10
3 5
5 10
5
1 2
3 3
4 3
5 3
30 5
```

Output for Sample Input

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

Problem K
Leap

Source file: `leap.{c | cpp | java}`
Input file: `leap.in`

A leap year is a year containing an extra day. In Georgian Calendar, the extra day is February 29th. The extra day is added to prevent the calendar year from drifting and keeps it synced with the astronomical and seasonal year.

This year, 2016, is a leap year. On February 29th of this year, there were discussions on some radio stations asking people who were born on February 29th to call in to wish them a happy birthday, because, as they put it, people born on this day celebrate their birthdays only once every 4 years. This is not a very accurate statement, as there are cases where someone might have to wait up to 8 years to celebrate their birthday.

A year is a leap year if it's divisible by 4, unless it's divisible by 100 (centurial years). Centurial years are only leap years if they are divisible by 400. For example, 2000 is a leap year, but 1900 is not a leap year.

In this problem, your task is to find out how many birthdays someone who's born on a leap day (February 29th) has had.

Input

The input starts with a number T ($0 < T < 1{,}000$) that represents the number of test cases in the file. Each test case consists of one line that contains two integers S and E ($0 < S \leq E \leq 10^{15}$) representing the birth year of the person born on a leap day, and the current year, respectively. Your calculations should include the current year.

Output

The output for each test case is in this form:

```
k. ans
```

where ***k*** represents the test case number (starting at 1), and ***ans*** is the number of birthdays the person has had up to and including the current given year. If the birth year is not a leap year, print "Not a leap year!" without the quotes, instead.

Sample Input	Output for Sample Input
3 2008 2016 1996 2020 1990 2024	1. 2 2. 6 3. Not a leap year!