2nd Gulf Programming Contest

Organized by

Khalifa University of Science, Technology and Research

Abu Dhabi

March 14th and 15th 2012

Problem Set

Duration: 9 am - 2 pm

Problem#	Problem Name	Balloon Colour
A	Word Shadow	Red
B	Gulf Tour	Pale Blue
C	Electrician Ladder	Lime Green
D	Robot Blip's Treasure World	Pink
E	Hands on GPS	Purple
F	Gannibals	white
G	Guess the Winner	Orange
H	Triangular Mesh Coloring	Yellow
I	Extermination Number List	Black
J	Facebook	Silver

Problem A
Word Shadow

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

In reality, when we read an English word we normally do not read every single letter of that word but rather the word's "shadow" recalls its pronunciation and meaning from our brain. The word's *shadow* consists of the same number of letters that compose the actual word with first and last letters (of the word) in their original positions while the rest of letters are interchanged; for example the sentence "I love programming" still can be read if it is written as "I lvoe pnimmargorg". Your task is to write a program that reads a set of sentences each of which is composed of a number of words; for each sentence, the program should produce another sentence which contains the exact number of words in their shadow form.

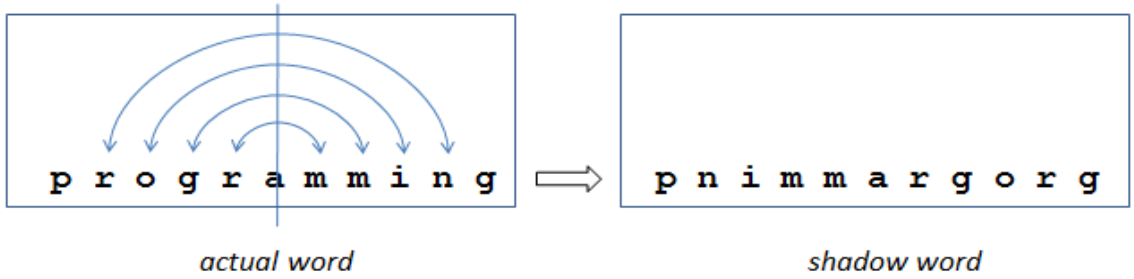


Figure 1: Example of conversion of a word to its shadow word

Input

In a single execution, the program should run for "T" test cases. The first line of the input file contains the integer variable T where $1 \leq T \leq 1000$. The rest of the T lines that follow, each contains two pieces of information about one sentence; the number of words (W) in the sentence and the text of the sentence itself. The variable W ranges from 1 to 10 inclusive. Each word may contain up to 12 letters. There is exactly one space between any two consecutive words in a sentence and sentences are entirely in small letters and do not contain any punctuation mark or special characters.

Output

Your program should produce for each test case, a sentence written in *shadow* format; words that are composed of less than four letters do not have *shadow* and if the number of letters composing a word is odd and more than four, the position of the central letter remains unchanged.

Sample Input

```
2
7 if anything can go wrong it will
8 a thing of beauty is a joy forever
```

Sample Output

```
if anihtyng can go wnorg it wlil
a tnihg of btuaey is a joy feveror
```

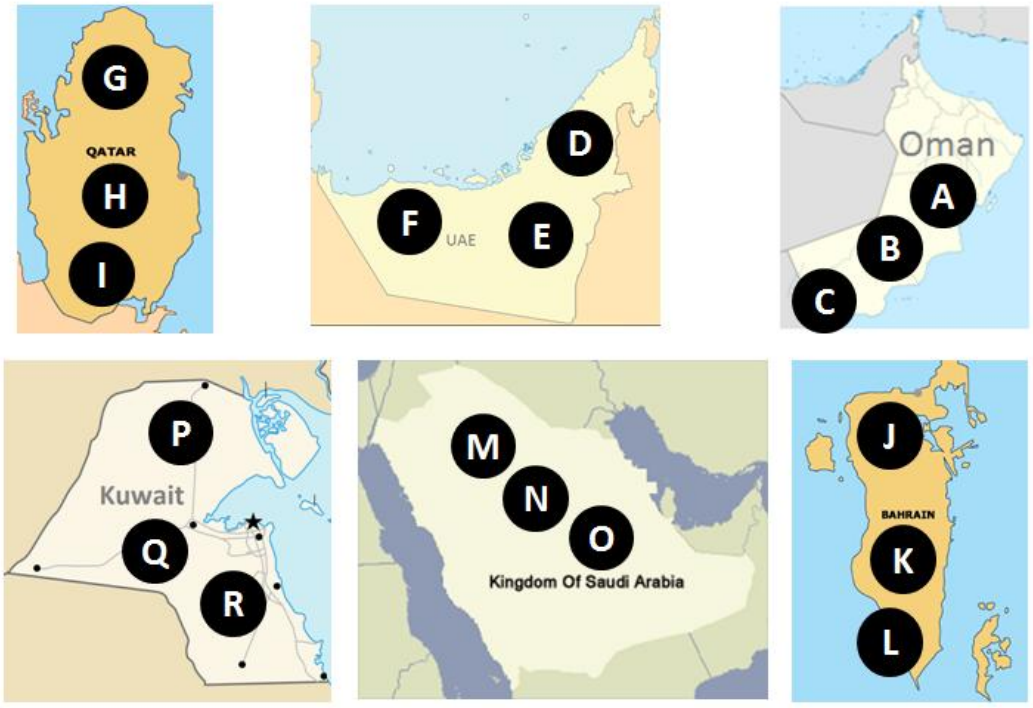
Problem B
Gulf Tour

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

A voyager has decided to discover the beauty of the Arabic GCC countries by making a shuttle round trip around the six states (Oman, UAE, Qatar, Bahrain, KSA and Kuwait). Due her limited time available for the entire voyage, the voyager must travel in the shortest path that links the six states starting and ending in the same country.

The Ministry of the Tourism at the departure state has provided the voyager with a list of THREE local airports at each state; eighteen airports around the six states. The list also contains the distance between each pair in the airports list. For simplicity, the rows and columns headers of the distance grid are named with letters from A to R where each successive THREE letters respectively represent three airports in Oman, UAE, Qatar, Bahrain, KSA and Kuwait. The voyager must visit exactly one airport at each state. The voyager must land exactly at one airport of each country in order to complete her round trip.

Your task is to write a program that reads the above described distance grid and helps the voyager to discover the shortest path to complete her round trip across the six states.



Input:

In a single execution, the program should run for T test cases. The first line of the input file contains the integer variable T where $1 \le T \le 10$. The input for each test case is specified as follows:

- The first line contains a letter corresponding to the airport from which the journey starts and ends.
- The following line starts with an asterisk "*" followed by alphabetic letters from A to R representing departure airports to destinations listed in the 1st column of the distance grid; all symbols, in the grid, are separated by at least one space.

- The distance grid is provided in 18 separate lines each of which starts with an alphabetic (ranging from A to R) and followed by 18 integers (the intersection of rows and columns) that represent the distance between airports. The given distances range from 0 to 9 units.

Output:

For each test case, your program should display the shortest path that crosses the six states and its length in one line of output composed of one integer and seven letters. The integer number is the sum of the distances between the airports that are included in the shortest path. The seven letters are the airports' codes that are included in the shortest path where the first and the last letters are identical as they represent the starting and ending airport of the journey. There is exactly one space between any two successive symbols in all output lines.

Sample Input

```
1
E
 *  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R
 A  0  1  3  1  3  6  5  3  4  6  5  1  7  6  1  9  6  8
 B     0  2  4  7  5  2  9  7  3  9  7  5  2  8  7  5  2
 C        0  9  2  8  1  8  6  4  2  8  3  4  9  4  1  3
 D           0  2  3  5  3  6  2  6  4  8  6  4  7  3  8
 E              0  1  1  4  9  9  3  5  5  7  3  1  4  6
 F                 0  2  8  7  7  1  8  9  2  1  2  5  9
 G                    0  3  2  2  6  4  8  6  7  8  7  6
 H                       0  1  1  3  9  5  9  2  1  4  3
 I                          0  5  8  7  4  1  3  2  9  5
 J                             0  2  3  6  5  3  7  2  4
 K                                0  1  1  2  7  3  6  8
 L                                   0  9  4  8  9  1  5
 M                                      0  1  2  2  7  6
 N                                         0  3  5  1  9
 O                                            0  3  8  4
 P                                               0  3  2
 Q                                                  0  1
 R                                                     0
```

Sample Output

```
15 E G C Q L N E
```

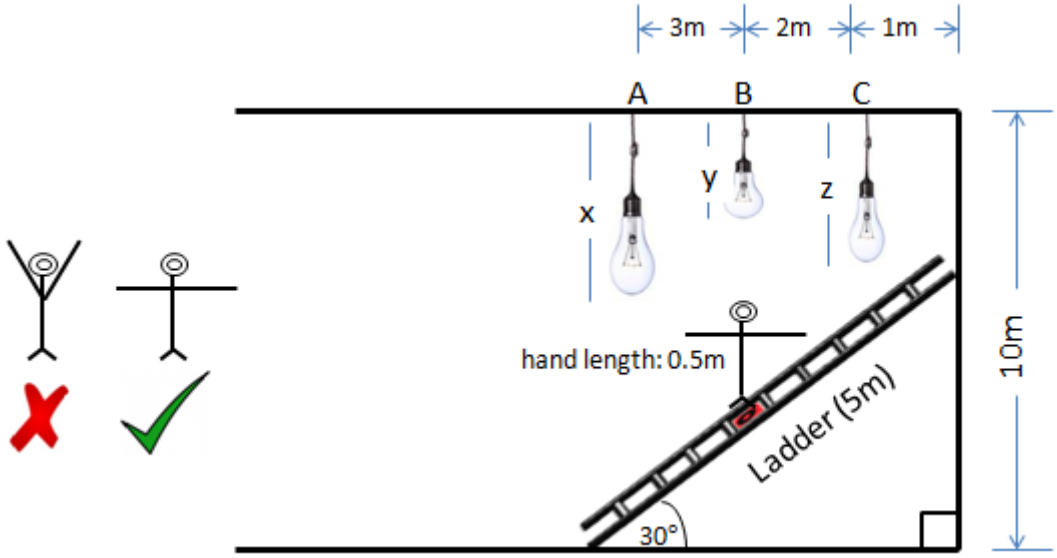
Problem C
Electrician Ladder

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

You are to help an electrician (shown in the picture below) to climb a ladder in order to reach a specific bulb that need to be fixed. The electrician uses a ladder of 5 meters in length laid against the vertical wall with an angle of 30°. The ladder has NINE steps distributed at equal distances from each other. The distance between the bottom-edge of the ladder and the first step is equivalent to the distance between any two successive steps. The ninth step is also located at a similar distance from the top-edge.

The electrician (his height up to shoulders = 1.67 meters, arm length = 0.5 meter; i.e. *two-arms* = 2×0.5 = 1.0 meter) has to fix THREE bulbs; A, B and C. The bulbs drop down from the ceiling at different heights (x, y and z respectively) in each test case. In addition, the bulbs A, B, and C are hanged at 3 meters, 2 meters and 1 meter away from the vertical wall respectively. The height of the vertical wall is 10 meters from the land-base.

Assuming that the electrician can ONLY expand his hands horizontally (i.e. rising hands up is not allowed), you are required to count the minimum number of steps that the electrician needs to climb in order to reach each bulb.



Input

In a single execution, the program should run for T test cases; the first line of the input file contains the integer variable T where 1 ≤ T ≤ 10000 and the remaining T lines each of which contains the length of the ropes x, y and z in centimeters separated by spaces.

Output

For each test case the program should display the minimum number of steps to be climbed by the electrician in order to reach each bulb. A line of output must contain three entries separated by a single space. An entry can be either a number (between 1 and 9) to indicate the minimum number of steps climbed to reach A, B and C respectively; or the letter N to indicate that a bulb cannot be reached.

Sample Input

```
10
51 738 770
24 107 807
484 151 601
89 658 141
890 813 712
455 360 793
372 202 276
43 173 866
651 875 828
532 861 69
```

Sample Output

```
N 4 6
N N 6
N N 7
N N N
1 4 6
N N 6
N N N
N N 6
N 4 6
N 4 N
```

Problem D
Robot Blip's Treasure World

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

Robot Blip lives in a two-dimensional world. The dimensions of the world are N x N glips, where 1 glip is a unit of length. The world is divided into 1x1 glip cells. Each cell in the world has a coordinate (x, y) (0 <= x, y < N). The robot easily fits into a cell. Robot Blip can accept three move commands: Left, Right, and Forward. If the command is Left or Right, then the Robot Blip will stay in the same cell but change its direction to left or right. If the command is Forward, Robot Blip will move forward to one cell in the direction he is heading to. Direction can be East, North, West, or South.

For example, if the current position of Robot Blip is (3, 2) and a Forward command is executed, his new position will be (4, 2), (3, 1), (2, 2), or (3, 3), if the direction is East, North, West, or South, respectively. By the way, (0, 0) is the North-West corner, and (N-1, N-1) is the South-East corner of the World.

Each cell in the world may also contain some energy. Depending on the cell location, the energy might be +1 elip, -1 elip, or 0 elip, where 1 elip is 1 unit of energy. When Robot Blip moves forward to a cell having z elips of energy, Blip gains z elips of treasure points. Note that z can be negative, in which case Blip will lose z points. For example, if Robot Blip's current treasure is 100, and he moves forward into a cell containing -1 (+1) elip energy, his treasure point will become 99 (101). However, Left and Right commands do not accrue any point.

Robot Blip is free to move into any cell, but not allowed to move beyond the World, which is of course within (0, 0) and (N-1, N-1). If Robot Blip is in a boundary cell (i.e., one of its coordinate is 0 or N-1), and commanded to move Forward beyond the World, it will simply ignore the command, but will lose 10 treasure points and its direction will be opposite. For example, if Blip's current position is (0, 5), and it faces West, and a Forward command is given, Blip will remain in (0, 5), but will face East now. If ever Blip's treasure point becomes negative, it will die and will no more accept commands.

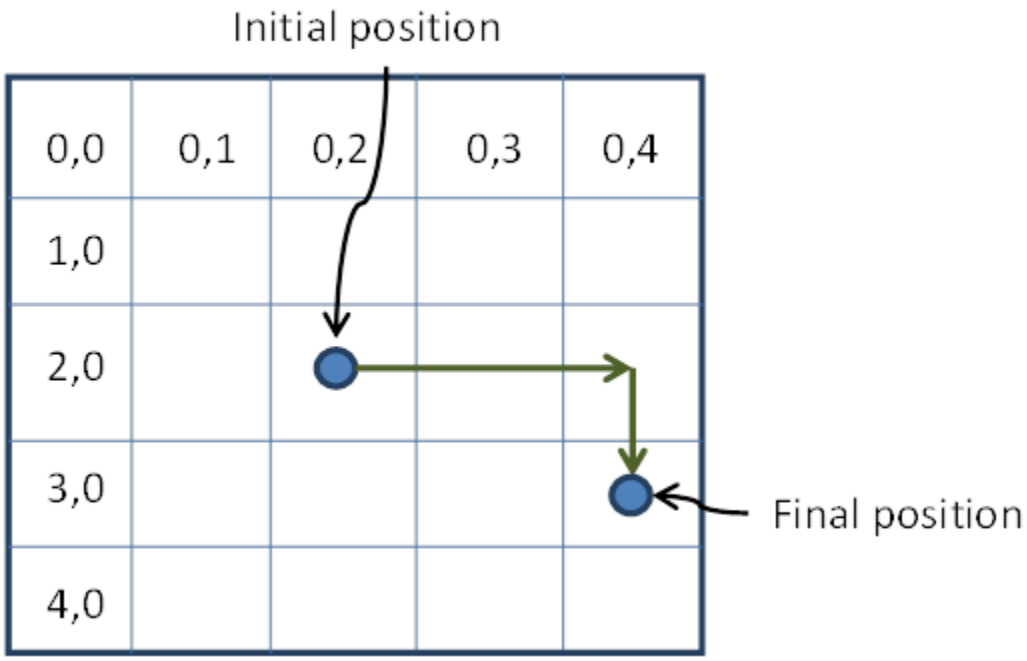


Figure 2: Movement of the robot in World #1 of the sample input

Input

The first line of input will contain a number C (0 < C <= 100) denoting the number of worlds to be processed; each world is subsequently specified as follows:

- On a separate line, a number N (0 < N <= 100) followed by N lines describing the World in a matrix form where $Line_i$ (0 <= i < N) describes the energy of the cells in the i^{th} row of the World; all integers are separated from each other by a space.
- The matrix is followed by a line of input containing an integer R, (0 <= R <= 100) denoting the number of cases to be processed for this world followed by R lines where each line contains three integers x, y, and t separated by a space, followed by a string D, followed by a space and a string S.
    - x and y denote the initial position (0 <= x, y < N) of Robot Blip,
    - t (t > 0) denotes the initial treasure points of Blip
    - D ∈ {"East", "North", "West", "South"} denotes the initial direction of Robot Blip.
    - S denotes the command string (maximum length = 100) in which each character denotes a command c_i ∈ {'L', 'R', 'F'}, where L=Left, R=Right and F=forward.

Output

For each world w first print the line "World #w", and the for each case i of the world, print one line "Case<space>#i:<space>x<space>y<space>t<space>D<space>S", where (x, y) is the final position of Robot Blip, t is the final treasure points that Blip has, D is the final direction (East/North/West/South). S is either Dead or Alive depending on whether Robot Blip died or not.

Sample Input

```
2
5
1 -1 1 1 -1
-1 1 1 -1 1
1 1 1 -1 -1
1 0 0 1 1
0 1 -1 1 1
1
2 2 20 East FFRFL
2
0 -1
1 1
1
0 0 1 East FFRL
```

Sample Output

```
World #1
Case #1: 3 4 19 East Alive
World #2
Case #1: 0 1 -10 West Dead
```

Problem E
Hands on GPS

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

Khaled's GPS lost signal in the desert; what a disaster! He is already late returning his rental car and he will have to pay 50 fills extra for every minute from now. For travelling each kilometer he also has to pay 20 fills and again there are some toll roads where each kilometer will cost an extra 10 fills. Without a GPS, Khaled doesn't know the most economical way to get the rental car returned. Luckily, he has the detailed map loaded in his laptop. Can you help Khaled to find the least cost path from his current position to the rental return location?

Input

The first input line consists of C (0 < C <= 100), the number of cases. For each case, the first line consists of two integers n (<=100) and m (<=10000), separated by a space, where n is the number of road intersections and m is the number of road segments. This is followed by m lines, where each line contains the description of a road segment as follows: S1<space>S2 <space>D<space>M<space>T, where,

- S1 and S2 (S1≠S¬2) are two integers denoting road intersection numbers (0 <=S1, S2 < n); S1 and S2 define the road segment and there is at most one road segment between them.
- D is a real number (D > 0) denoting the length of the road segment in Kilometers
- M is the time in minutes (M > 0) required to travel the road segment
- T is a string, which is either "True" or "False" depending on whether the road segment is a toll road or not. All road segments are bidirectional.

For each case, the intersection numbers of Khaled and the rental car return are specified after the m lines specifying road segments on a separate line of input consisting of two integers A, B (0 <= A, B < n); A denotes Khaled's intersection number and B denotes the rental car return's intersection number and A≠B.

Output

For each case, first print the case number and then the time (t), cost (c) in fills, distance (d) in kilometers for the most economic route, as follows: Case<space>#:t<space>c<space>d. Cost, time and distance are to be rounded to two decimal places.

Sample Input

```
1
4 5
0 1 2 5 False
1 2 1 2 False
0 2 3 2 False
1 3 3 1 False
2 3 9 7 True
0 3
```

Sample Output

```
Case #1: 5.00 390.00 7.00
```

Problem F
Gannibals

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

In a far distant cluster of Galaxies, an extraordinary event of Galactic cannibalism is taking place where large galaxies are swallowing up smaller galaxies. These galaxies are called Galactic cannibals, or *Gannibals,* in short. There are certain rules for swallowing:

- Galaxy X can swallow up another galaxy Y only if the galactic area (defined further down) of X is at least twice the galactic area of Y.
- After swallowing galaxy Y, the rank of galaxy X becomes 1+ maximum (rank of X, rank of Y).
- After swallowing Y, galaxy X cannot swallow any other galaxy, but X can be swallowed up by another galaxy Z if rule 1 is satisfied (i.e., galactic area of Z is at least twice the galactic area of X)

The initial rank of all the galaxies is zero. The rank of a cluster of galaxies is equal to the highest rank of the galaxies in the cluster. Each galaxy is known by three extreme peripheral stars of the galaxy, called p-stars. The galactic area is equal to the area of the circumcircle of the triangle defined by the p-stars. For those who are wondering how to compute the area of a circumcirle of a triangle, here is a hint: the radius *r* of the circumcircle of a triangle is: $r = \frac{abc}{4.A}$, where *a*, *b*, and *c* are the lengths of the three sides of the triangle, and *A* is the area of the triangle. Given a cluster of galaxies, the problem is to find the highest possible rank of the cluster.

Input

The first input line will consist of C (0 < C <= 100), the number of cases to process. Each case describes one cluster of galaxies. The first line of each case contains an integer k (0 < k <= 100), the number of galaxies in the cluster. This line is followed by k lines, each containing the description of one galaxy. The description of each galaxy consists of six integers (space separated) as follows: x1 y1 x2 y2 x3 y3, where (x_i, y_i) is the coordinate of the i^{th}p-star of the galaxy; all galaxies are assumed to be two dimensional. The three p-stars are guaranteed to be *not* co-linear.

Output

For each case, first output the case number and then the integer corresponding to the highest possible rank of the cluster as follows: Case #: r

Sample Input

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

Sample Output

```
1:2
2:4
```

Problem G
Guess the Winner

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

Ahmed and his friend love to play the following game: a set of stones are divided into m heaps, each heap contains a number of stones between 1 and n. The players alternate turns removing stones from a chosen heap (one heap only). In each turn a player can remove at least one stone from the heap. The last one to take a stone(s), leaving an empty heap is the winner. Here is an example with three heaps A, B and C containing initially three, five and nine stones:

A	B	C	Comment on the Move
3	5	9	Ahmed takes 3 from C
3	0	6	friend takes 5 from
3	0	5	Ahmed takes 1 from C
3	0	1	friend takes 4 from C
1	0	1	Ahmed takes 2 from A
1	0	0	friend takes 1 from C
0	0	0	Ahmed takes 1 from A and he wins

In this game Ahmed lets his friend set the initial configuration of the heaps but in return, he has the right of deciding who will make the first move. Ahmed knows that for a given initial configuration, he can guarantee to be the winner provided he takes the right decision on who start first. The number of heaps is fixed and equal to 10. The maximum number of stones per heap is 9.

Write a program to help Ahmed in his decision.

Input

The first line of input contains an integer N indicating the number of test cases (initial heap configurations), followed by N lines, each line contains 10 numbers representing the number of stones in each heap.

Output

For each test case, print, in a new line 1 if Ahmed should play first and 2 otherwise

Sample Input

```
15
6 4 9 2 5 9 9 6 3 9
6 2 9 3 3 5 9 1 2 8
9 1 9 8 2 5 7 3 5 6
```

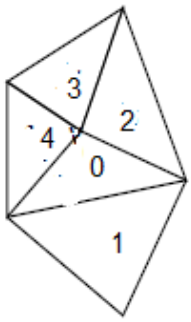
Sample Output

```
2
1
1
```

Problem H:
Triangular Mesh Coloring

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

In computer graphics 3D triangular mesh surface is a standard format for representing object shapes. Basically, a triangular mesh is a group of connected triangular facets that encodes a surface's shape in terms of geometry and connectivity. Each facet is labeled by a unique number and as shown in Figure 3, the facets connectivity is stored in an nX3 array, where n is the number of facets, and whereby the array's index corresponds to the facet's number. Each element in the array contains the indexes of the facet which are adjacent to that facet (sharing a common edge). -1 means that the facet has no adjacent facet on that edge. Some applications of computer graphics require a binary coloring of triangular mesh, that is, coloring the facets with two different colors (e.g. black and white), subject to the constraint that two adjacent facets must not have the same color. You are requested to write a program that performs this mesh binary coloring.



1	2	4
0	-1	-1
0	-1	3
2	-1	4
3	-1	0

Figure 3: Example

Input

The first line of input contains an integer N indicating the number of test cases, followed by series of test cases. The test case data starts with the number of facets followed by the adjacency array. You can assume that each facet has at least one adjacent facet, and that the maximum number of facets in each test case is 100.

Output

For each test case, print on the standard output, in a new line, the color code of each facet (0 for black and 1 for white). If the binary coloring is impossible then print -1. The first facet in the mesh should be colored in black. Outputs from different test cases outputs should be separated by the string *****.

Sample Input

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

Sample Output

```
0
1
1
0
1
*****
-1
```

Problem I

Extermination Number List
Time limit: 30 seconds

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

A student needs help to solve the following Mathematics assignment: he needs to compute the value f(x) as defined in the Figure below; the number x is defined such that -100000 ≤ x ≤ 100000.

Your job is to calculate f(x) for each x read from the input file.

$$f(x) = \begin{cases} f(x-1)+f(x-2)+7 & x>1 \text{ and prime} \\ f(x-1)-f(x-2)+4 & x>1 \text{ and not prime} \\ f(x+1)+f(x+2)-4 & x<0 \end{cases}$$

$$f(0)=12,\ f(1)=6$$

Figure: Definition of the function f

Input

The input is organized as follows. The first line of the input file includes an integer number N to indicate the number of test cases that follow, and each following line includes an integer number representing a test case x.

Output

The value f(x) corresponding to each input on a separate line.

Sample Input	Sample Output
6 1 0 -2 5 6 7	6 12 22 62 49 118

Problem J

Facebook

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

Facebook is a social networking service and website launched in February 2004. Users must register before using the site, after which they may create a personal profile, add other users as friends and exchange messages. Each person on Facebook has 0 or more friends. Given part of the Facebook network; a mini-network (consisting of a maximum of 26 members named A, B, C, …, Z).

You are required to find the number of common friends (if any) between any 2 of the Facebook members in the mini-network. A common friend of A and B is a direct friend of A and at the same time a direct friend of B.

Input

The first line of the input specifies the number of test cases (mini-networks) to be processed. Each test case starts with the number n of lines describing the mini-network in Facebook. A description line such as A<space>B means that A and B are direct friends. A positive integer m will immediately follow the description of the mini-network and it provides the number of queries to be answered; each query is on a separate line and is in the form of two letters separated by a space. For example the query B<space>D means how many common friends are there between B and D.

Output

For each query line there should be an output line specifying the number of common friends and the names of these common friends; all upper-case, separated by single spaces, and in alphabetical order.

Sample Input

```
1
8
A  B
A  D
C  D
E  F
F  E
C  E
D  F
B  C
4
A  F
B  F
E  F
B  D
```

Sample Output

```
1  D
0
0
2  A C
```