Code Jam Beta 2008

A. Triangle Trilemma
B. The Price is Wrong
C. Random Route
D. Hexagon Game

Questions asked

Submissions

Triangle Trilemma
10pt Not attempted **244/318 users** correct (77%)
10pt Not attempted **200/260 users** correct (77%)

The Price is Wrong
15pt Not attempted **110/175 users** correct (63%)
25pt Not attempted **67/96 users** correct (70%)

Random Route
30pt Not attempted **42/76 users** correct (55%)
30pt Not attempted **38/51 users** correct (75%)

Hexagon Game
25pt Not attempted **8/29 users** correct (28%)
45pt Not attempted **6/15 users** correct (40%)

Top Scores

malcin	190
marek.cygan	190
SnapDragon	165
ardiankp	145
Astein	130
rem	130
RAVEman	120
yuhch123	120
Lovro	120
lukasP	120

Problem A. Triangle Trilemma

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 10 points	Solve A-small
Large input 10 points	Solve A-large

Problem

You're interested in writing a program to classify triangles. Triangles can be classified according to their internal angles. If one of the internal angles is exactly 90 degrees, then that triangle is known as a "right" triangle. If one of the internal angles is greater than 90 degrees, that triangle is known as an "obtuse" triangle. Otherwise, all the internal angles are less than 90 degrees and the triangle is known as an "acute" triangle.

Triangles can also be classified according to the relative lengths of their sides. In a "scalene" triangle, all three sides have different lengths. In an "isosceles" triangle, two of the sides are of equal length. (If all three sides have the same length, the triangle is known as an "equilateral" triangle, but you can ignore this case since there will be no equilateral triangles in the input data.)

Your program must determine, for each set of three points, whether or not those points form a triangle. If the three points are not distinct, or the three points are collinear, then those points do not form a valid triangle. (Another way is to calculate the area of the triangle; valid triangles must have non-zero area.) Otherwise, your program will classify the triangle as one of "acute", "obtuse", or "right", and one of "isosceles" or "scalene".

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each case is a line formatted as

```
x1 y1 x2 y2 x3 y3
```

Output

For each test case, output one line containing "Case #**x**: " followed by one of these strings:

- `isosceles acute triangle`
- `isosceles obtuse triangle`
- `isosceles right triangle`
- `scalene acute triangle`
- `scalene obtuse triangle`
- `scalene right triangle`
- `not a triangle`

Limits

1 ≤ **N** ≤ 100,
x1, **y1**, **x2**, **y2**, **x3**, **y3** will be integers.

Small dataset

0 ≤ **x1**, **y1**, **x2**, **y2**, **x3**, **y3** ≤ 9

Large dataset

-1000 ≤ **x1**, **y1**, **x2**, **y2**, **x3**, **y3** ≤ 1000

Sample

Input	Output
8 0 0 0 4 1 2 1 1 1 4 3 2 2 2 2 4 4 3 3 3 3 4 5 3 4 4 4 5 5 6 5 5 5 6 6 5 6 6 6 7 6 8 7 7 7 7 7 7	Case #1: isosceles obtuse triangle Case #2: scalene acute triangle Case #3: isosceles acute triangle Case #4: scalene right triangle Case #5: scalene obtuse triangle Case #6: isosceles right triangle Case #7: not a triangle Case #8: not a triangle

Problem B. The Price is Wrong

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input
15 points
Solve B-small

Large input
25 points
Solve B-large

Problem

You're playing a game in which you try to guess the correct retail price of various products for sale. After guessing the price of each product in a list, you are shown the same list of products sorted by their actual prices, from least to most expensive. (No two products cost the same amount.) Based on this ordering, you are given a single chance to change one or more of your guesses.

Your program should output the smallest set of products such that, if you change your prices for those products, the ordering of your guesses will be consistent with the correct ordering of the product list. The products in the returned set should be listed in alphabetical order. If there are multiple smallest sets, output the set which occurs first lexicographically.

For example, assume these are your initial guesses:

- `code = $20`
- `jam = $15`
- `foo = $40`
- `bar = $30`
- `google = $60`

If the correct ordering is `code jam foo bar google`, then you would need to change two of your prices in order to match the correct ordering. You might change one guess to read `jam = $30` and another guess to read `bar = $50`, which would match the correct ordering and produce the output set `bar jam`. However, the output set `bar code` comes before `bar jam` lexicographically, and you can match the correct ordering by changing your guesses for these items as well.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each case consists of two lines. The first line contains the list of products, space separated, sorted from least to most expensive. Each product's name will consist of only lowercase letters and underscores. There will be no duplicate products in the list. The second line contains your initial guesses for each product in the list, respectively. All guesses will be integers between 1 and 100, inclusive. The number of guesses will be equal to the number of products, and no two guesses will be the same. The guesses will be space separated. (Note that although the initial guesses all happen to be integers, you are allowed to change your guesses to any amounts, not just integers.)

Output

For each test case, output one line containing "Case #**x**: " followed by the set of products for which you must change your prices. The products should be in alphabetical order and space separated. The constraints guarantee that you will need to change the price of at least one product for each test case.

Limits

1 ≤ **N** ≤ 100.

Small dataset

2 ≤ number of products in list ≤ 8

Large dataset

2 ≤ number of products in list ≤ 64

Sample

Input	Output

<pre>1
code jam foo bar google
20 15 40 30 60</pre>
	<pre>Case #1: bar code</pre>

Code Jam Beta 2008

A. Triangle Trilemma
B. The Price is Wrong
C. Random Route
D. Hexagon Game

Questions asked

Submissions

Triangle Trilemma

10pt	Not attempted **244/318 users** correct (77%)
10pt	Not attempted **200/260 users** correct (77%)

The Price is Wrong

15pt	Not attempted **110/175 users** correct (63%)
25pt	Not attempted **67/96 users** correct (70%)

Random Route

30pt	Not attempted **42/76 users** correct (55%)
30pt	Not attempted **38/51 users** correct (75%)

Hexagon Game

25pt	Not attempted **8/29 users** correct (28%)
45pt	Not attempted **6/15 users** correct (40%)

Top Scores

malcin	190
marek.cygan	190
SnapDragon	165
ardiankp	145
Astein	130
rem	130
RAVEman	120
yuhch123	120
Lovro	120
lukasP	120

Problem C. Random Route

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 30 points	Solve C-small
Large input 30 points	Solve C-large

Problem

Where do you want to go today, and how do you want to get there? You decide to choose the answer to both questions at random.

You will be given a list of roads. Each road connects one city to another city (all roads are one-way), and each road takes a certain amount of time to drive. You will also be given a starting city. Consider all cities that you're able to drive to, not including your starting city, and choose one of them at random with uniform probability to be your destination. Now consider every fastest route from your starting city to your destination city, and choose one of these routes at random with uniform probability. This will be the route on which you end up driving.

For each road in the input, your program must output the probability that you will end up driving on that road, given the behavior outlined above.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each case begins with a line formatted as

```
num_roads starting_city
```

This will be followed by **num_roads** lines, each formatted as

```
city1 city2 time
```

Each line represents a one-way road that starts at `city1` and ends at `city2`, and takes `time` hours to drive. All cities will be formatted as strings consisting of only lowercase letters and underscores. For each road, `city1` will not be equal to `city2`, and `time` will be an integer between 1 and 100000, inclusive. The starting city is guaranteed to appear as `city1` on at least one road; therefore, there will always be at least one possible destination (and at least one shortest route to that destination).

Output

For each test case, output one line containing "Case #**x**: " followed by the probability that you will drive on each road, in the same order that the roads were listed in the input. Probabilities should be space separated and formatted so there are exactly seven digits after the decimal point. Each probability must be within a distance of 1e-6 from the correct answer to be judged as correct.

Limits

$1 \leq \mathbf{N} \leq 100$.

Small dataset

$2 \leq$ **num_roads** ≤ 25.

Large dataset

$2 \leq$ **num_roads** ≤ 50.

Sample

Input	Output

<pre>1
5 san_francisco
san_francisco los_angeles 6
los_angeles san_diego 2
san_francisco san_diego 8
los_angeles san_diego 2
san_francisco los_angeles 6</pre>
	Case #1: 0.4500000 0.2000000

Code Jam Beta 2008

A. Triangle Trilemma

B. The Price is Wrong

C. Random Route

D. Hexagon Game

Questions asked

Submissions

Triangle Trilemma

10pt	Not attempted **244/318 users** correct (77%)
10pt	Not attempted **200/260 users** correct (77%)

The Price is Wrong

15pt	Not attempted **110/175 users** correct (63%)
25pt	Not attempted **67/96 users** correct (70%)

Random Route

30pt	Not attempted **42/76 users** correct (55%)
30pt	Not attempted **38/51 users** correct (75%)

Hexagon Game

25pt	Not attempted **8/29 users** correct (28%)
45pt	Not attempted **6/15 users** correct (40%)

Top Scores

malcin	190
marek.cygan	190
SnapDragon	165
ardiankp	145
Astein	130
rem	130
RAVEman	120
yuhch123	120
Lovro	120
lukasP	120

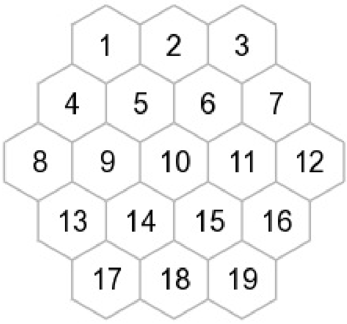
Problem D. Hexagon Game

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 25 points	Solve D-small
Large input 45 points	Solve D-large

Problem

You are playing a game on a hexagonal board of size **S**. The middle row is composed of **S** hexagons, and the top and bottom rows each have (**S** + 1) / 2 hexagons. (**S** will be odd.) The hexagons are numbered starting with 1 in the upper left, and increasing left-to-right and top-to-bottom. Here is a hexagonal board of size **S**=5:



The game starts with **S** checkers on the board. Multiple checkers might start in the same position. Each checker also has an associated integer value between 0 and 50, inclusive. A turn consists of choosing a checker and moving it to an adjacent position, which increments your score by the value of that checker. Checkers cannot move off the board. Each position can contain any number of checkers at the same time.

The game ends when all the checkers are lined up in a straight row, with exactly one checker per hexagon. There are three possible ending configurations on any board. For **S**=5, the game will end when checkers are in positions (1, 5, 10, 15, 19), or in positions (3, 6, 10, 14, 17), or in positions (8, 9, 10, 11, 12). Your program must output the smallest possible score of a finished game.

For example, assume the checkers start in positions (1, 2, 5, 15, 19). The checker in position 1 has a value of 1, the checkers in positions 2 and 5 have values of 3, and the checkers in positions 15 and 19 have values of 0. You could move the checker from position 1 into position 5 and then position 10. Both of these moves add one point to your score. Then you could move the checker from position 2 into position 1, adding three points to your score. This game would end with a score of 5, which is the lowest possible score for this starting configuration.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each case consists of two lines. The first line contains the starting positions of the checkers, space separated. The second line contains the values of each checker, respectively, space separated.

Output

For each test case, output one line containing "Case #**x**: " followed by the minimum possible score of a finished game.

Limits

1 ≤ **N** ≤ 100.

Small dataset

3 ≤ **S** ≤ 15; **S** is odd.

Large dataset

3 ≤ **S** ≤ 75; **S** is odd.

Sample

Input	Output

```
1               Case #1: 5
1 2 5 15 19
1 3 3 0 0
```