IDI Open
Warmup Contest
April 10th, 2010

The Problem set

A	Addition (Easy)
B	Hansel and Grethel
C	Theta Puzzle
D	Nth Largest Value (Easy)
E	Black Knight
F	The Picnic
G	Stacking Slabs
H	Markov Trains
I	Euro Efficiency
J	Balls

Tips

- Tear the problem set apart and share the problems among you.
- Problems are not ordered by difficulty.
- Try solving the easy problems first. Two problems in this set are tagged with "(Easy)" to help point you in the right direction.
- If your solution fails on a problem, you can print your program and debug it on paper while you let someone else work on a different problem on the computer.
- If you need help, contact the judges.

Rules

- Each team consists of one to three contestants.
- One computer is used per team.
- You may not cooperate with persons not on your team.
- You may print your programs on paper to debug them.
- What you may bring to the contest:
  - Any written material (Books, manuals, handwritten notes, printed notes, etc).
  - Pens, pencils, blank paper, stapler and other useful non-electronic office equipment.
  - NO material in electronic form (CDs, USB pen and so on).
  - NO electronic devices (PDAs and so on).
- The only electronic content you may consult during the content is that specified by the organiser (see the web-page). You may not copy source code from web pages, etc.
- Your programs should read from standard in and write to standard out. Writing to standard error will result in a failed submission. C programs should return 0 from `main()`.
- Your program may use at most 100MB of memory.
- Your programs may not:
  - access the network,
  - read or write files on the system,
  - talk to other processes,
  - fork,
  - or similar stuff.
  - If you try, your program will hang or crash. If it hangs, it will take a couple of minutes before others will be able to run their programs. And please do not crack somebody who uses their spare time trying to give you something valuable.
- Show common sense and good sportsmanship.

Problem A

Addition (Easy)

Your little brother is having a hard time learning to add. As an effort to ensure he never learns this, you decide to write a program that adds two numbers for him.

Input specifications

The first line of input contains a single number T , the number of test cases. The following T lines contain two numbers each, A and B, the numbers you need to add.

Output specifications

For each test case, print a single line containing the number that equals A + B.

Notes and Constraints

- $0 < T \leq 100$
- $-1000 \leq A, B \leq 1000$

Sample input

```
2
1 3
-7 0
```

Output for sample input

```

4
-7
```

Problem B

Hansel and Grethel

On a warm summer afternoon, Hansel and Grethel are walking together in the fields. It is getting late and, to be honest, they are lost. Grethel is a little scared, still vividly remembering the last time they got lost in the forest. That time, an evil witch had locked them inside a house built of gingerbread and sugar! But Hansel can reassure her: this time they are well prepared. Hansel has taken a map and a compass with him!

Hansel picks two clearly outstanding features in the landscape, and uses the compass to measure the direction towards both objects. Grethel locates the objects on the map, and writes down the corresponding map coordinates. Based on this information, they will be able to accurately determine their own position on the map.

The coordinates of two marker objects, and the direction (angle from the North) towards these objects are known. Write a program which uses this data to calculate the coordinates of Hansel and Grethel's current location.

Input specifications

The first line of the input contains T, the number of test cases that follow. Then follow two lines per test case, describing the two marker objects. Each marker object is described by a line containing three integer numbers, the x-coordinate x of the object on the map, the y-coordinate y of the object on the map and the direction d of the object in degrees.

Output specifications

Output one line per test case containing the result of the position calculation: two numbers, separated by a space, each having exactly 4 digits after the decimal point. These numbers represent the x and y coordinates of the position of Hansel and Grethel (0 x y 100). Round the numbers as usual: up if the next digit would be 5, down otherwise.

Notes and Constraints

- $0 \leq x, y \leq 100$
- $0 \leq d < 360$ with with 0 being North, 90 being East, 180 being South, and so on.
- The x-axis runs West-to-East on the map, with increasing values towards the East.
- The y-axis runs South-to-North on the map, with increasing values towards the North.
- The directions of the two objects are not exactly equal, and do not differ by exactly 180 degrees.

Sample input

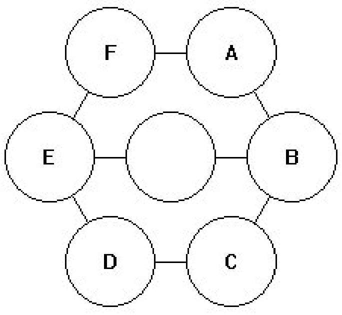
```
2
30 50 90
20 40 180
30 40 96
20 20 150
```

Output for sample input

```
20.0000 50.0000
7.0610 42.4110
```

Problem C

Theta Puzzle



The Theta Puzzle consists of a base with 6 positions at the vertices of a regular hexagon and another position at the center, connected as shown in the figure. There are six tokens labeled A, B, C, D, E and F. A single move of the puzzle is to move a token to an adjacent empty position (along an allowed connection - the line segments in the diagram below). The idea of the puzzle is to start with an initial arrangement of tokens with the center empty and, by a sequence of moves, get to the configuration in the figure.

An initial position for the puzzle is given by a permutation of the letters A through F. The first letter starts at A in the figure, the next at B and so on.

Write a program which, given an initial permutation, either finds the shortest sequence of moves to solve the puzzle or determines that there is no solution.

Input specifications

The first line of input contains a single integer T, the number of test cases that follow. Each test case consists of a single line containing a permutation of the letters A through F giving the initial puzzle position.

Output specifications

For each test case, output a single line containing the minimum number of moves to solve the puzzle. If there is no solution, output `NO SOLUTION` instead.

Notes and Constraints

- $0 < T \leq 1000$)

Sample input

```
12
FACDBE
ABCDEF
ADCEFB
ADCEBF
FEDCBA
FEDCAB
ECBFAD
ECBFDA
DCEBFA
DCEBAF
CBEADF
BDEAFC
```

Output for sample input

```
5
0
19
NO SOLUTION
29
NO SOLUTION
19
NO SOLUTION
13
NO SOLUTION
21
16
```

Problem D

Nth Largest Value (Easy)

For this problem, you will write a program that prints the Nth largest value in an array of integers. To make things simple, N will be 3 and the array will always be have 10 decimal integer values.

Input specifications

The first line of input contains a single integer T, the number of test cases that follow. Each data set consists of a single line containing 10 space separated decimal integers, a_i.

Output specifications

For each data set, output a single line containing the 3rd largest value of the 10 integers.

Notes and Constraints

- $0 < T \leq 1000$
- $0 < a_i \leq 1000$

Sample input

```
4
1 2 3 4 5 6 7 8 9 1000
338 304 619 95 343 496 489 116 98 127
931 240 986 894 826 640 965 833 136 138
940 955 364 188 133 254 501 122 768 408
```

Output for sample input

```
8
489
931
768
```

Problem E

Black Knight

Every time you visit your uncle Per, he gives you a new puzzle to test you. This time, you find a chess board and a single knight on the table. Your challenge tonight is to move the knight as many times as possible on a given part of the chessboard, without using a square twice. Per says he will give you 50 kroner if you can move the knight more than 20 times on a 5x5 square.

You find this puzzle to be time consuming and boring. Luckily, your other uncle, Truls, has taught you that cheating is not immoral before you are caught. So while pretending to be texting your girlfriend, you write a program on your cell phone solving the problem.

A knight moves two squares vertically and one horisontally, or one vertically and two horisontally.

Input specifications

The first line of input gives t, the number of test scenarios. Each scenario is given by a line with four integers m, n, i and j, where m and n give the size of the usable board, and i and j give a starting position on the board.

Output specifications

For each test case, print the maximal number of jumps you can do with a knight on the given board, never using the same square twice.

Notes and Constraints

- $1 \leq t \leq 100$
- $1 \leq m \leq 8$
- $1 \leq n \leq 8$
- $m \cdot n \leq 25$
- $0 \leq i < m$
- $0 \leq j < n$

Sample input

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

Output for sample input

```
7
0
10
```

Problem F

The Picnic

The annual picnic of the Zeron company will take place tomorrow. This year they have agreed on the Gloomwood park as the place to be. The girl responsible for the arrangement, Lilith, thinks it would be nice if everyone is able to watch everyone else during the occasion. From geometry class she remembers that a region in the plane with the property that a straight line between any two points in the region, lies entirely in the region, is called convex. So that is what she is looking for. Unfortunately, this seems hard to fulfill, since Gloomwood has many opaque obstacles, such as large trees, rocks, and so on.

Owing to the fact that the staff of the Zeron company is pretty large, Lilith has a rather intricate problem to solve: finding a location to hold them all. Therefore, some of her friends help her to draw a map of the whereabouts of the largest obstacles. To mark out the place, she will use a ribbon stretched around the obstacles on the circumference of the chosen region. The opaque obstacles should be thought of as points of zero extension.

Input specifications

The first line of the input contains a single integer T, the number of test cases that follow. Each test case consists of two lines. The first one contains an integer m, the number of obstacles in the park . The next line contains $2m$ integers, the coordinates of the m obstacles, in the order $x_1y_1x_2y_2x_3y_3$...

Output specifications

For each test case, output a single line containing the area of the largest convex polygon having obstacles as corners, but no enclosed obstacles, with one decimal.

Notes and Constraints

- $2 < m < 100)$
- $0 \leq x_i, y_i \leq 1000$
- Each scenario has at least three obstacles that are not on a straight line, and no two obstacles have the same coordinates.

Sample input

```
1
11
3 3 8 4 12 2 22 3 23 5 24 7 27 12 18 12 13 13 6 10 9 6
```

Output for sample input

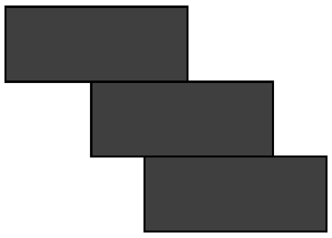
```
129.0
```

Problem G

Stacking Slabs

Jarle has just started in the first grade, studying civil engineering. Unfortunately for his peace of mind, his girlfriend Bjørgfrid studies architecture. This week she got an assignment at school, where she is to design the stairs to a building. A part of the "specification" is that they must consist of identical slabs, stacked one on top of the previous, without gluing or bolting them together.

Fig 1



The stairs are to pass over a body of water, with lillies growing around it. Now Bjørgfrid ask Jarle how many slabs she needs to pass the water. He asks her how many persons the stairs should support, but the "specification" does not say, so they assume none.

A slab is 1 x 1 x 0.1 m, and has a density of 2400 kg/m^3.

Input specifications

The first line of input gives $1 \leq n \leq 100$, the number of test cases. Then follow n lines, each with a floating-point number $0.1 < d < 2.1$, giving the distance to be passed by the stairs in each test case. There will be no cases where a minimal number of slabs *exactly* pass the distance.

Output specifications

For each test case, output the number of slabs needed to pass the distance, or "`impossible`" if it cannot be done.

Sample input

```
2
0.4999
0.74999
```

Output for sample input

```

2
3
```

Problem H

Markov Trains

Business is not going well for the Dutch Railway Company NS. Due to technical problems, they are forced to cancel many train services without advance notice. This is, of course, extremely frustrating for students who travel from home to school by train.

The worst thing about the whole situation is the randomness of the cancellations. Nobody knows in advance whether a train service will be canceled; a cancellation is not announced until the official departure time. Since there is usually more than one possible route from home to school, people are often left with an 'if I had known this in advance I would have taken the other route' sort of feeling.

Recently, the statistics department of the NS found a revolutionary solution to this problem. They noticed that some train services are canceled more often than others. In order to help the passengers, they decided to publish this information. The new timetables will state not just the time of departure and arrival of each service, but also its probability of cancellation.

The travel-planner software from the NS, which normally finds the fastest route between stations, must be updated to find the route which gives the best chance of arriving in time. This helps passengers to avoid trains that are likely to cause problems, instead taking a slightly longer, but more reliable route to school.

Given the new timetables, a departure station and time, a destination station and a desired arrival time, find the route which gives the best chance of arriving at the destination in time.

A route in this case is simply an ordered list of stations visited by the passenger, starting with the departure station and ending with the destination. The passenger will stick to the route, each time taking the first possible train to the next station. If a train is canceled, he will just wait for the next train to that station.

The chance of arriving in time is taken to be the probability that the passenger, when following the route as described above, arrives at the destination station before or at the desired arrival time.

Input specifications

The first line of the input contains a single positive integer indicating the number of runs. For each run, the input is as follows:

- A line with a single positive integer n, the number of trains in the timetable.
- n lines describing the timetable. Each line describes one train, stating its departure station x, the time of departure t_x, its destination station y, the time of arrival t_y and its probability of cancellation p.
- A line with the travellers departure station a, earliest departure time t_a , destination station b and desired arrival time t_b.

Output specifications

The output consists a single line, containing the probability that the passenger, when following the optimal route, arrives on time. The probability must be formatted as a decimal real number with exactly one digit before the decimal point, and exactly 4 digits after.

Notes and Constraints

- $0 < T \leq 200$
- $0 \leq n \leq 100$
- In the time table, $x \neq y$ and $t_x < t_y$, while for the traveller $a \neq b$ and $t_a < t_b$
- Stations are identified by capital letters in the range `'A'-'L'`. Times are in the format $hh : mm$ with $00 \leq hh < 24$ and $00 \leq mm < 60$.
- $0.0 \leq p \leq 1.0$
- The usual rules for rounding apply: round up if the next digit would be 5 or greater, otherwise round down.
- You can safely assume that train services are canceled independently of each other and according to the probabilities stated in the timetable.
- When changing trains at an intermediate station, the earliest possible departure time is one minute after the time of arrival.
- All times are on the same day; the journey does not cross midnight.
- It never happens that two or more trains depart from the same station at the same time to the same destination station.
- The input is such that there is a unique route with maximum probability.
- The passenger will stick to his route, always taking the first available train to the next station. If a train is cancelled he will wait for the next train to that station. He will never try to be smart by taking faster trains or different routes.

Sample input

```
2
3
A 12:00 B 12:15 0.1
A 12:10 B 12:14 0.23
A 12:20 B 12:30 0.456
A 12:00 B 12:30
4
A 12:00 B 12:15 0.1
A 12:05 B 12:13 0.15
B 12:20 C 12:35 0.12
A 12:15 C 12:33 0.4
A 12:00 C 13:00
```

Output for sample input

```
0.9895
0.8668
```

Problem I

Euro Efficiency

On January 1st 2002, The Netherlands, and several other European countries abandoned their national currency in favour of the Euro. This changed the ease of paying, and not just internationally.

A student buying a 68 guilder book before January 1st could pay for the book with one 50 guilder banknote and two 10 guilder banknotes, receiving two guilders in change. In short, $50 + 10 + 10 - 1 - 1 = 68$. Other ways of paying were $50 + 25 - 5 - 1 - 1$ or $100 - 25 - 5 - 1 - 1$. Either way, there are always 5 units (banknotes or coins) involved in the payment process, and it could not be done with less than 5 units.

Buying a 68 Euro book is easier these days: $50 + 20 - 2 = 68$, so only 3 units are involved. This is no coincidence; in many other cases paying with euros is more efficient than paying with guilders. On average the Euro is more efficient. This has nothing to do, of course, with the value of the Euro, but with the units chosen. The units for guilders used to be: 1, 2 1/2, 5, 10, 25, 50, whereas the units for the Euro are: 1, 2, 5, 10, 20, 50.

For this problem we restrict ourselves to amounts up to 100 cents. The Euro has coins with values 1, 2, 5, 10, 20, 50 eurocents. In paying an arbitrary amount in the range [1, 100] eurocents, on average 2.96 coins are involved, either as payment or as change. The Euro series is not optimal in this sense. With coins 1, 24, 34, 39, 46, 50 an amount of 68 cents can be paid using two coins. The average number of coins involved in paying an amount in the range [1, 100] is 2.52.

Calculations with the latter series are more complex, however. That is, mental calculations. These calculations could easily be programmed in any mobile phone, which nearly everybody carries around nowadays. Preparing for the future, a committee of the European Central Bank is studying the efficiency of series of coins, to find the most efficient series for amounts up to 100 eurocents. They need your help.

Write a program that, given a series of coins, calculates the average and maximum number of coins needed to pay any amount up to and including 100 cents. You may assume that both parties involved have sufficient numbers of any coin at their disposal.

Input specifications

The first line of the input contains T, the number of test cases. Each test case is described by 6 different positive integers on a single line: the values of the coins, in ascending order. The first number is always 1. The last number is less than 100.

Output specifications

For each test case, output a single line containing first the average and then the maximum number of coins involved in paying an amount in the range [1, 100]. These values are separated by a space. As in the example, the average should always contain two digits behind the decimal point. The maximum is always an integer.

Sample input

```
3
1 2 5 10 20 50
1 24 34 39 46 50
1 2 3 7 19 72
```

Output for sample input

```
2.96 5
2.52 3
2.80 4
```

Problem J

Balls

The classic Two Glass Balls brain-teaser is often posed as:

> "Given two identical glass spheres, you would like to determine the lowest floor in a 100-story building from which they will break when dropped. Assume the spheres are undamaged when dropped below this point. What is the strategy that will minimize the worst-case scenario for number of drops?"

Suppose that we had only one ball. We'd have to drop from each floor from 1 to 100 in sequence, requiring 100 drops in the worst case.

Now consider the case where we have two balls. Suppose we drop the first ball from floor n. If it breaks we're in the case where we have one ball remaining and we need to drop from floors 1 to $n-1$ in sequence, yielding n drops in the worst case (the first ball is dropped once, the second at most $n-1$ times). However, if it does not break when dropped from floor n, we have reduced the problem to dropping from floors $n+1$ to 100. In either case we must keep in mind that we've already used one drop. So the minimum number of drops, in the worst case, is the minimum over all n.

You will write a program to determine the minimum number of drops required, in the worst case, given B balls and an M-story building.

Input specifications

The first line of input contains a single integer T, the number of test cases that follow. Each data set consists of a single line containing 2 integers; the number of balls B and the number of floors in the building M.

Output specifications

For each data set, output one line containing the minimum number of drops needed in the worst case for the corresponding numbers B and M.

Notes and Constraints

- $0 < T \leq 1000$
- $0 < B \leq 50$
- $0 < M \leq 1000$

Sample input

```
4
2 10
2 100
2 300
25 900
```

Output for sample input

```
4
14
24
10
```