CARROLL SENIOR HIGH SCHOOL
Computer Science Competition 2017

*Artwork by Peyton Rodgers

CARROLL SENIOR HIGH SCHOOL

Computer Science Competition
Hands-On Programming Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the format exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Problems

Number	Name
Problem 1	Hospital
Problem 2	Trap101
Problem 3	Lucky
Problem 4	Taxi
Problem 5	Pigeon
Problem 6	Weight
Problem 7	Paints
Problem 8	Sticky
Problem 9	Loading
Problem 10	Stack
Problem 11	Chore
Problem 12	Counting

1. Hospital

Program Name: Hospital.java **Input File: hospital.dat**

Kevin wants to visit all of the children in the New York Children's Hospital this Christmas. To make things easier, write a program to make a list of all the rooms he has to visit.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of two integers x and y, denoting the number of floors, and the number of rooms on each floor, respectively.

Output

For each floor and room number, print the following on its own line : "`floor x room y`". Print a blank line between test cases.

Example Input File

```
3
2 3
1 5
5 1
```

Example Output to Screen

```
floor 1 room 1
floor 1 room 2
floor 1 room 3
floor 2 room 1
floor 2 room 2
floor 2 room 3

floor 1 room 1
floor 1 room 2
floor 1 room 3
floor 1 room 4
floor 1 room 5

floor 1 room 1
floor 2 room 1
floor 3 room 1
floor 4 room 1
floor 5 room 1
```

2. Trap101

Program Name: Trap101.java **Input File: trap101.dat**

Kevin is setting a trap for the Wet Bandits with his brother's weight lifting equipment. He needs to balance circular weights on the bar so that it won't fall over until he triggers the trap. He needs help determining if a given configuration of weights will allow the trap to work or not.

$$\text{Center of Mass} = \frac{m_1 x_1 + m_2 x_2 + \ldots + m_n x_n}{m_1 + m_2 + \ldots + m_n}$$

Input

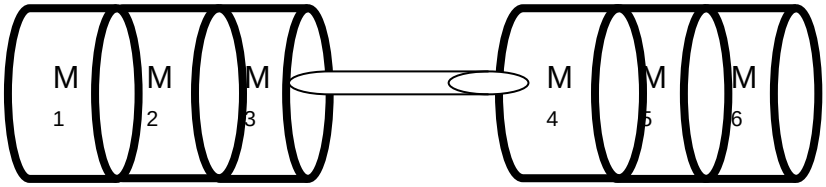
The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of three lines. The first line of each data set will consist of two integers `x` and `y`, representing the number of weights on the left and right ends of the bar, respectively. The next two lines will consist of x and y integers, respectively, representing how heavy each weight is, starting on each side with the weight closest to the center of the bar.

Output

If the bar has the same amount on both sides, and the center of mass lies exactly in the center of the bar, print "`works`". Otherwise, print "`doesn't work`"

Example Input File

```
4
4 4
1 2 3 4
1 2 3 4
3 2
1 2 3
5 1
4 4
1 2 3 2
1 1 5 1
3 3
1 2 3
1 3 1
```



Example Output to Screen

```
works
doesn't work
works
doesn't work
```

3. Lucky

Program Name: Lucky.java **Input File: lucky.dat**

The Wet Bandits are playing a counting game to decide who gets to deal with Kevin when they find him. The person who says their name when Kevin is found is lucky. Write a program to show what happens each count until Kevin is found.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of three integers `x`, `y`, and `k`. `x` represents the number of counts between each "`Marv`". `y` represents the number of counts between each "`Harry`". `k` is the number of counts before they find Kevin.

Output

For each test case, starting at count 1 and continuing until count `k`, print "`Marv`" if Marv is lucky, "`Harry`" if Harry is lucky, and "`MarvHarry`" if they are both lucky. If neither of them are lucky, simply print the count. Each output should appear on its own line, and a blank line should be left between test cases.

Example Input File

```
2
2 3 9
1 2 4
```

Example Output to Screen

```
1
Marv
Harry
Marv
5
MarvHarry
7
Marv
Harry

Marv
MarvHarry
Marv
MarvHarry
```

4. Taxi

Program Name: Taxi.java **Input File: taxi.dat**

Kevin is outside of his hotel in New York. He notices something interesting about the people waiting in line for a taxi. After some time, people decide to abandon the line rather than continuing to wait. Given the intervals of time it takes for each person to get a taxi, determine how many people get a taxi and how many abandon the line.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with a single integer `x` denoting how many total people try to get a taxi. The next `x` lines of each test case will consist of two unique integers, a and b, denoting the start and end of each time interval people will wait, respectively. No two people will arrive at or leave the line at the same time. Those who arrive at the line with no one else ahead of them will still use their whole interval. When one person gets a taxi, the next person to get a taxi is whoever arrived the earliest out of those still there. If a person's interval ends and they haven't reached the front of the line, they simply abandon the line and give up getting a taxi.

For example, given the example input below, the first person arrives at t = 1, and stays until they get a taxi at t = 5. The second person arrives at t = 2 and gets in line until the first person gets in a taxi at t = 5, then takes until t = 10 to get a taxi. The next person arrives at t = 3 and will stay in line until t = 7, when they abandon the line because the second person's interval isn't complete.

Output

For each test case on its own line, output in the format "x y z", x being the number of people to get a taxi, y being the number of people to abandon the line, and z being the total amount of time people were waiting in line.

Example Input File

```
1
10
1 5
2 10
3 7
4 8
6 12
9 14
11 20
13 16
15 18
17 19
```

Example Output to Screen

```
5 5 19
```

5. Pigeon

Program Name: Pigeon.java **Input File: pigeon.dat**

The pigeon lady loves to feed pigeons. Each day, some of the pigeons who show up form friendships with each other, and if the pigeon sees that pigeon friend on a later day, they proceed to eat out of the same hand. This allows all of a pigeon's friends to meet with every other friend. For instance, if pigeon A and B were friends, as well as pigeon C and D, a friendship between B and C would cause the group to expand to size 4. Given a list of friendships formed, write a program to determine the largest group of friends, as well as the number of pigeons who remained loners throughout the day.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with two integers, k and `x,` denoting how many pigeons visited the lady that day and how many friendships were formed. The next x lines of each data set will consist of two integers y and z (1 <= y < z <= k), representing a friendship forming between pigeon y and pigeon z.

Output

For each test case, print two lines of the following format:
"`largest group: x`
`loners: y`".
leave a blank line between test cases

Example Input File

```
2
10 4
1 2
2 3
3 4
4 5
15 7
1 2
3 4
5 6
7 8
2 5
4 8
1 8
```

Example Output to Screen

```
largest group: 5
loners: 5

largest group: 8
loners: 7
```

6. Weight

Program Name: Weight.java **Input File: weight.dat**

While setting up another trap, Kevin finds a mystery weight in Buzz's room. It is unmarked and he can't seem to discern the weight by picking it up. He's able to accurately measure the density, and has measured the dimensions of the weight as well. The weight is circular with a hole in the middle for a bar to go through. Given the dimensions and density of the weight, can you help Kevin determine its weight?

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of 4 integers `d`, r_1, r_2, and `h`, denoting the density of the weight in kg/cm^3, the radius of the weight in cm, the radius of the hole in the center in cm, and the height (thickness) in cm.

Output

For each test case on its own line, output the mass of the weight in kg, rounded to three decimal places.

Example Input File

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

Example Output to Screen

```
94.248
62.832
28.274
```

7. Paints

Program Name: Paints.java **Input File: paints.dat**

Kevin, being the tricky dude he is, has made a trap that will spill paint on the floor of a square room. He wants to know how much of the floor will be covered in paint after he activates the trap. Each bucket will spill in a perfect circle around a point in the room. Write a program to find the area of the room that is covered in paint as well as the area of the floor that remains paint-free.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with two integers, s and k, denoting the length of one side of the room and the number of paint cans. The next k lines will each consist of three integers, x and y, and r, (0 <= x <= s, 0 <= y <= s, r <= s), denoting the position of the center of each splatter of paint and the radius.

Output

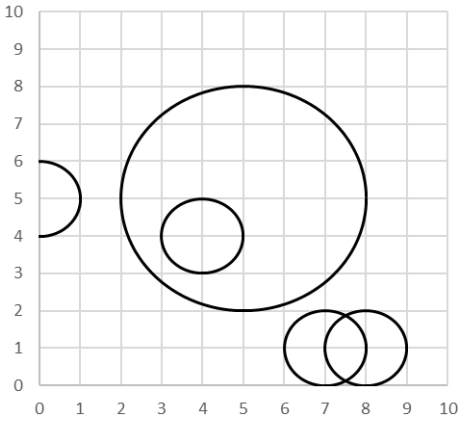
For each test case on its own line, output the approximate area Kevin has covered and how much he has left to cover to definitely catch the Wet Bandits, each rounded to 1 decimal place.

Example Input File

```
1
10 5
0 5 1
5 5 3
4 6 1
7 1 1
8 1 1
```

Example Output to Screen

```
34.9 65.1
```



8. Sticky

Program Name: Sticky.java **Input File: sticky.dat**

Kevin is planning a sticky trap, and he'd like to cover as much area as possible. His strips of sticky substance are the same width as the trap, but he has to worry about the length. Find the maximum number of strips he can fit into his trap without any overlapping.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with a single integer `x` denoting how many sticky substance strips he has (say that five times fast!). The following `x` lines of each data set will consist of two integers, s and `e`, denoting the beginning and ending position where each strip must be applied. (It's a very temperamental trap, and each strip can only be applied at certain positions.) Two strips can be applied without conflicting as long as one ends at or before the other begins.

Output

For each test case on its own line, print the maximum number of sticky substances Kevin can place on his trap.

Example Input File

```
1
10
1 5
2 10
3 7
4 8
6 12
9 14
11 20
13 16
15 18
17 19
```

Example Output to Screen

```
4
```

9. Loading

Program Name: Loading.java **Input File: loading.dat**

After a few failed attempts at setting his weight trap, Kevin has decided that he will only load weights symmetrically in order to decrease the work he has to do to determine if a configuration of weights is possible. For example, if he puts a 5 pound weight on one side, he will also put a 5 on the other side in the same position. He has a limited number of weights, and in addition to this, only a certain number of weights will fit on the bar. For example, he can't try to fit 20 one pound weights on the bar. Given his collection of weights, the capacity of the bar, and the desired weight to load on the bar, determine whether or not it is possible to load the bar to the desired weight.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with 3 integers `x`, `y`, and `z` denoting how many weight types there are, the weight Kevin wants on the bar, and the number of weights the bar can hold on each side. The next two lines will each consist of x integers, denoting the mass of each weight, and how many there are of each, respectively.

Output

For each test case on its own line, print "possible" if it is possible to properly load the bar with exactly Kevin's desired weight. Otherwise, print "not possible".

Example Input File

```
3
2 40 5
5 4
4 5
1 10 4
1
10
4 40 5
10 5 2 1
2 2 4 10
```

Example Output to Screen

```
not possible
not possible
possible
```

10. Stack

Program Name: Stack.java **Input File: stack.dat**

Kevin is bored at home, so he decides to stack his Christmas cookies. He only has five left, and each is a different size. They are arranged in two stacks. He wonders how much effort it would take to arrange the cookies such that they were all on one stack, and that all cookies were sorted with the largest sizes on the bottom. He is able to pick up any set of cookies from the top of a stack, but expends effort equal to the total of the cookie sizes of that group. Determine the effort required to arrange the cookies in one stack if he acts optimally.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with two integers x and y, denoting the number of cookies on the first and second stack. The next two lines will consist of x and y integers, respectively, and will represent the cookies on each stack, ordered from bottom to top.

Output

For each test case on its own line, output the minimum effort required to arrange all cookies on one stack in sorted order.

Example Input File

```
1
4 1
1 2 4 3
5
```

Example Output to Screen

```
10
```

11. Chore

Program Name: Chore.java **Input File: chore.dat**

Kevin doesn't want to go outside while Old Man Marley is outside. He observes him for several hours and notices that what chore he is doing outside depends on what he was doing the previous hour. For instance, he would unlikely to be shoveling his driveway directly after shoveling his driveway the previous hour. Kevin has managed to determine the probability of going between every pair of chores from one hour to the next. In addition to this, Kevin knows what Marley was doing the previous hour. Given this information, help Kevin determine the probability of Old Man Marley doing each type of chore.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with a single integer `x` denoting how many different chores Old Man Marley does. The next `x` lines will each consist of `x` floating point numbers between 0 and 1. The `y`th number of line `x` is the probability of doing chore `y` the hour after chore `x`, and the sum of each row will be 1. The last line of each test case will consist of an integer `k` ($0 <= k < x$) and an integer `d`, representing which chore Old Man Marley did last time Kevin saw him, and how many hours ago this was, respectively. Each chore is mutually exclusive, e.g. Old Man Marley will not do two chores in the same hour.

Output

For each test case on its own line, print the probability of Old Man Marley doing chore 0 through chore `x`-1 at this hour in a space delimited list. Round all answers to three decimal places.

Example Input File

```
1
3
0.5 0.2 0.3
0.1 0.6 0.3
0.2 0.7 0.1
0 2
```

Example Output to Screen

```
0.330 0.430 0.240
```

12. Counting

Program Name: Counting.java **Input File: counting.dat**

Kevin needs to buy some chemicals to defeat Harry and Marv. He calculates the amount of mercury, arsenic, and hydrogen peroxide he needs and heads to the store. Each box at the toxin store has a certain amount of each of the three toxins, as well as a price. What is the cheapest way to reach his required amounts? Keep in mind that he may have to go over for some chemicals to meet all of his needs.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with four integers, denoting how many different boxes there are, and how much mercury, arsenic, and hydrogen peroxide you need, respectively. The next x lines each consist of a price (floating point with exactly two decimal places) followed by 3 integers denoting the mercury, arsenic, and hydrogen peroxide content of each box. The store has only one of each box.

Output

For each test case on its own line, output the minimum cost of satisfying Kevin's needs, or print "not possible" if there are not enough chemicals at the store.

Example Input File

```
2
6 13 10 6
1.00 5 4 3
1.00 7 2 1
1.00 1 2 3
0.50 6 1 4
0.75 3 2 1
0.8 2 5 5
6 30 10 6
1.00 5 4 3
1.00 7 2 1
1.00 1 2 3
0.50 6 1 4
0.75 3 2 1
0.8 2 5 5
```

Example Output to Screen

```
2.30
not possible
```

Credits

Artwork by Peyton Rodgers (2017)

Problems from A+ Computer Science (www.apluscompsci.com)

"Kevin" theme supplied by Carroll Computer Science students,
based on "Home Alone" movies from John Hughes and Twentieth Century Fox