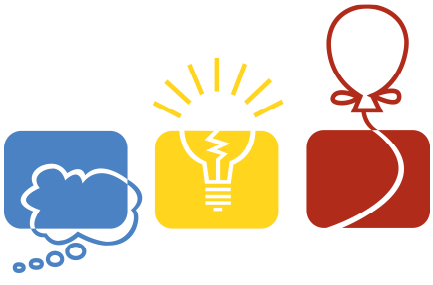


IBM event sponsor

Programming Day,

31. Oktober 2010

A 11296 - Counting Solutions to an Integral Equation

Time limit: 6.000 seconds

Problem

Given, **n**, count the number of solutions to the equation **x+2y+2z=n**, where **x,y,z,n** are non negative integers.

The Input

There is at most 1500 inputs. Each input is **n** (**n**<1000001) on a single line.

The Output

For each input, output the number of solutions on a single line.

Sample Input

```
2
3
```

Sample Output

```
3
3
```

B 11243 Texas Trip

After a day trip with his friend Dick, Harry noticed a strange pattern of tiny holes in the door of his SUV. The local American Tire store sells fiberglass patching material only in square sheets. What is the smallest patch that Harry needs to fix his door?

Assume that the holes are points on the integer lattice in the plane. Your job is to find the area of the smallest square that will cover all the holes.

Input

The first line of input contains a single integer T expressed in decimal with no leading zeroes, denoting the number of test cases to follow. The subsequent lines of input describe the test cases.

Each test case begins with a single line, containing a single integer n expressed in decimal with no leading zeroes, the number of points to follow; each of the following n lines contains two integers x and y, both expressed in decimal with no leading zeroes, giving the coordinates of one of your points.

You are guaranteed that T <= 30 and that no data set contains more than 30 points. All points in each data set will be no more than 500 units away from (0,0).

Output

Print, on a single line with two decimal places of precision, the area of the smallest square containing all of your points. An answer will be accepted if it lies within 0.1 of the correct answer.

Sample Input

```
2
4
-1 -1
1 -1
1 1
-1 1
4
10 1
10 -1
-10 1
-10 -1
```

Sample Output

```
4.00
242.00
```

C 10149 - Yahtzee

Time limit: 3.000 seconds

The game of Yahtzee involves 5 dice, which are thrown in 13 rounds. A score card contains 13 categories; each round may be scored in a category of the player's choosing, but each category may be scored only once in the game. The 13 categories are scored as follows:

- ones - sum of all ones thrown
- twos - sum of all twos thrown
- threes - sum of all threes thrown
- fours - sum of all fours thrown
- fives - sum of all fives thrown
- sixes - sum of all sixes thrown
- chance - sum of all dice
- three of a kind - sum of all dice, provided at least three have same value
- four of a kind - sum of all dice, provided at least four have same value
- five of a kind - 50 points, provided all five dice have same value
- short straight - 25 points, provided four of the dice form a sequence (that is, 1,2,3,4 or 2,3,4,5 or 3,4,5,6)
- long straight - 35 points, provided all dice form a sequence (1,2,3,4,5 or 2,3,4,5,6)
- full house - 40 points, provided three of the dice are equal and the other two dice are also equal. (for example, 2,2,5,5,5)

Each of the last six categories may be scored as 0 if the criteria are not met.

The score for the game is the sum of all 13 categories plus a bonus of 35 points if the sum of the first six categores (ones to sixes) is 63 or greater.

Your job is to compute the best possible score for a sequence of rounds.

The Input

Each line of input contains 5 integers between 1 and 6, indicating the values of the five dice thrown in each round. There are 13 such lines for each game, and there may be any number of games in the input data.

The Output

Your output should consist of a single line for each game containing 15 numbers: the score in each category (in the order given), the bonus score (0 or 35), and the total score. If there is more than categorization that yields the same total score, any one will do.

Sample Input

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

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 1 1 1 1
6 6 6 6 6
6 6 6 1 1
1 1 1 2 2
1 1 1 2 3
1 2 3 4 5
1 2 3 4 6
6 1 2 6 6
1 4 5 5 5
5 5 5 5 6
4 4 4 5 6
3 1 3 6 3
2 2 2 4 6
```

Output for Sample Input

```
1 2 3 4 5 0 15 0 0 0 25 35 0 0 90
3 6 9 12 15 30 21 20 26 50 25 35 40 35 327
```

D 10032 - Tug of War

Time limit: 3.000 seconds

A tug of war is to be arranged at the local office picnic. For the tug of war, the picnickers must be divided into two teams. Each person must be on one team or the other; the number of people on the two teams must not differ by more than 1; the total weight of the people on each team should be as nearly equal as possible.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line of input contains *n* the number of people at the picnic. *n* lines follow. The first line gives the weight of person 1; the second the weight of person 2; and so on. Each weight is an integer between 1 and 450. There are at most 100 people at the picnic.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Your output will be a single line containing 2 numbers: the total weight of the people on one team, and the total weight of the people on the other team. If these numbers differ, give the lesser first.

Sample Input

```
1

3
100
90
200
```

Sample Output

```
190 200
```

E 10040 - Ouroboros Snake

Time limit: 3.000 seconds

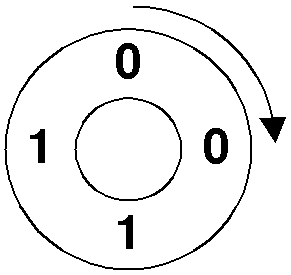
Background

Ouroboros was a mythical snake in Ancient Egypt. It has its tail inside its mouth and continuously devours itself.

Problem

Ouroboros numbers are binary numbers of 2^n bits that have the property of generating the whole set of numbers from 0 to 2^n-1 as follows: To produce all of them we place the 2^n bits wrapped in a circle so that the last bit goes before the first one. Then we can denote all 2^n different strings with n bits starting each time with the next bit in the circle.

For example, for n=2 there are only four Ouroboros numbers. These are 0011, 0110, 1100 and 1001. For 0011, the following diagram and table depicts the process of finding all the bitstrings:



k	**0011**0011...	*o*(*n*=2,*k*)
0	00	0
1	01	1
2	11	3
3	10	2

Your program will compute the function ***o*(*n*,*k*)**, where ***n* > 0** and $0 \leq k < 2^n$. This function calculates the ***k***-th number inside the smallest Ouroboros number of size ***n***-bits.

Input

The input starts with a line containing the number of test cases. For each test case you will be given a line with two integers n ($0<n<22$) and k ($0 \leq k < 2^n$).

Output

For every test case your output must evaluate the function $o(n,k)$ and print the result on a line by itself.

Sample Input

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

Sample Output

```
0
1
3
2
```

F 10056 - What is the Probability?

Time limit: 3.000 seconds

Probability has always been an integrated part of computer algorithms. Where the deterministic algorithms have failed to solve a problem in short time, probabilistic algorithms have come to the rescue. In this problem we are not dealing with any probabilistic algorithm. We will just try to determine the winning probability of a certain player.

A game is played by throwing a dice like thing (it should not be assumed that it has six sides like an ordinary dice). If a certain event occurs when a player throws the dice (such as getting a 3, getting green side on top or whatever) he is declared the winner. There can be N such player. So the first player will throw the dice, then the second and at last the N th player and again the first player and so on. When a player gets the desired event he or she is declared winner and playing stops. You will have to determine the winning probability of one (The I th) of these players.

Input

Input will contain an integer S (S<=1000) at first, which indicates how many sets of inputs are there. The next S lines will contain S sets of inputs. Each line contain an integer N (N<=1000) which denotes the number players, a floating point number p which indicates the probability of the happening of a successful event in a single throw (If success means getting 3 then p is the probability of getting 3 in a single throw. For a normal dice the probability of getting 3 is 1/6), and I (I<=N) the serial of the player whose winning probability is to be determined (Serial no varies from 1 to N). You can assume that no invalid probability (p) value will be given as input.

Output

For each set of input, output in a single line the probability of the I th player to win. The output floating point number will always have four digits after the decimal point as shown in the sample output.

Sample Input:

```
2
2 0.166666 1
2 0.166666 2
```

Sample Output:

```
0.5455
0.4545
```

G 11343 Isolated Segments

Time Limit: 1 sec , Memory Limit: 16MB

You're given n segments in the rectangular coordinate system. The segments are defined by start and end points (X_i,Y_i) and (X_j,Y_j) (1 <= i, j <= n). Coordinates of these points are integer numbers with real value smaller then 1000. Length of each segment is positive.

When 2 segments don't have a common point then it is said that segments don't collide. In any other case segments collide. Be aware that segments collide even if they have only one point in common.

Segment is said to be isolated if it doesn't collide with all the other segments that are given, i.e. segment i is isolated when for each 1 <= j <= n, (i != j), segments i and j don`t collide. You are asked to find number T - how many segments are isolated.

INPUT:

First line of input contains number N (N <= 50), then tests follow. First line of each test case contains number M (M <= 100) - the number of segments for this test case to be considered. For this particular test case M lines follow each containing a description of one segment. Segment is described by 2 points: start point (X_{pi},Y_{pi}) and end point (X_{ei},Y_{ei}). They are given in such order: X_{pi} Y_{pi} X_{ei} Y_{ei}

OUTPUT:

For each test case output one line containing number T.

SAMPLE INPUT:

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

SAMPLE OUTPUT:

```
1
0
2
0
0
```

H 10951 Polynomial GCD

Given two polynomials **f(x)** and **g(x)** in $\mathbf{Z_n}$, you have to find their **GCD** polynomial, ie, a polynomial **r(x)** (also in $\mathbf{Z_n}$) which has the greatest degree of all the polynomials in $\mathbf{Z_n}$ that divide both **f(x)** and **g(x)**. There can be more than one such polynomial, of which you are to find the one with a leading coefficient of **1** (**1** is the unity in $\mathbf{Z_n}$. Such polynomial is also called a *monic polynomial*).

(Note: A function **f(x)** is in $\mathbf{Z_n}$ means all the coefficients in **f(x)** is **modulo n**.)

Input

There will be no more than **101** test cases. Each test case consists of three lines: the first line has **n**, which will be a prime number not more than **1500**. The second and third lines give the two polynomials **f(x)** and **g(x)**. The polynomials are represented by first an integer **D** which represents the degree of the polynomial, followed by **(D + 1)** positive integers representing the coefficients of the polynomial. the coefficients are in decreasing order of Exponent. Input ends with **n = 0**. The value of **D** won't be more than **100**.

Output

For each test case, print the test case number and **r(x)**, in the same format as the input

Sample Input	Output for Sample Input
3 3 2 2 1 1 4 1 0 2 2 2 0	Case 1: 2 1 2 1

Note: The first sample input has $2x^3 + 2x^2 + x + 1$ *and* $x^4 + 2x^2 + 2x + 2$ *as the functions.*

I 11642 Bangladesh Premier League

So, here comes another cricket tournament – Bangladesh Premier League aka BPL. In this tournament, **N** teams play against each other in a round-robin league fashion for a total of **R** rounds. Team with highest number of wins is announced the winner of the tournament. If there are multiple teams with highest number of wins, then all of them are winners. Now, we are in the middle of a tournament and would like to determine which of the teams are mathematically out of the league race. In other words, we would like to determine which teams cannot win the tournament even if they win all of their remaining matches.

You are given the current league table of the tournament. For example, let's say there 4 teams in the tournament – Dhaka, Chittagong, Rajshahi and Khulna. A sample table can be like this –

Team	Wins	Loss	Left	Dhk	Ctg	Raj	Khl
Dhaka	6	7	5	0	0	0	5
Chittagong	10	4	4	0	0	4	0
Rajshahi	10	4	4	0	4	0	0
Khulna	1	12	5	5	0	0	0

From the table, you can see that Khulna is definitely not in the league race. They have won 1 match and can only win 5 more which would be insufficient to pass Chittagong or Rajshahi. On the other hand, Dhaka is also eliminated though it may not be evident from the table. Dhaka has won 6 matches and can win 5 more making a total of 11. Now Chittagong and Rajshahi have won a total of 10+10 = 20 matches and they will play each other 4 more times. So, on the average, they will win (20+4) / 2 = 12 matches. So, one team would definitely win more than 11 matches thereby eliminating Dhaka.

Input

Input will consist of less than 30 test cases. First line of each test case will contain no of teams in the tournament, N (1<N<51). Then there will be a table in the format mentioned above except for the headers. It will consist of N rows and N+4 columns. Each row will represent a team. Team names don't have any spaces in them and they will be at most 30 characters in length. Last N columns of the table constitute the 'Remaining Game Matrix' which is symmetrical and its diagonal entries are

always zero. No. of rounds, R (0<R<31) will not be explicitly mentioned in the statement. The last test case will be followed by a -1 in a line by itself, denoting the end of input file.

Output

For each test case, you should print "Case C:" in the first line where C is replaced by the case number starting from 1. Then, for each team eliminated, you should output a block of 5 lines which should be in the following format –

Team X is eliminated.

They can win at most a + b = c games.

T_1, T_2, … and T_n have won a total of G games.

They play each other M times.

So, on average, each of the teams wins (G+M)/n = p games which is greater than c.

Here,

X is the name of the team which is eliminated.

a, b and c are the no. of games team X has already won, no. of remaining games and their total wins optimistically respectively.

T_1, T_2, … and T_n are the n team names you think will mathematically beat team X as a combination. G is their total match won already. There will be M matches between these n teams.

Print a blank line after each such block. Print an extra blank line after each test case.

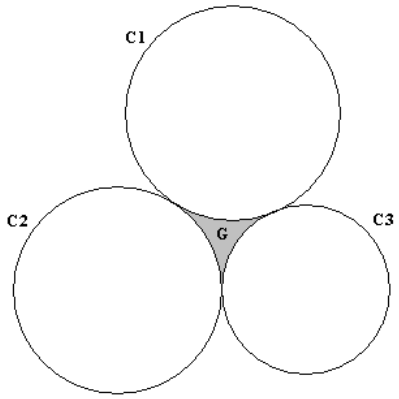
If there are multiple groups that satisfy the condition, you can output any one. Also you can output teams in any order. Output will be checked by a special judge program. For floating point number (only p), you can print any number of digits after the decimal point provided it does not differ by more than 1E-6 from the original value.

Sample Input	Output for Sample Input
4 A 6 7 5 0 0 0 5 B 10 4 4 0 0 4 0 C 10 4 4 0 4 0 0 D 1 12 5 5 0 0 0 2 A 1 1 1 0 1 B 1 1 1 1 0 -1	Case 1: Team A is eliminated. They can win at most 6 + 5 = 11 games. B and C have won a total of 20 games. They play each other 4 times. So, on average, each of the teams wins (20+4)/2 = 12.000000 games

	which is greater than 11. Team D is eliminated. They can win at most 1 + 5 = 6 games. B have won a total of 10 games. They play each other 0 times. So, on average, each of the teams wins (10+0)/1 = 10.000000 games which is greater than 6. Case 2: [the line above is blank]

Note: Though the output blocks are supposed to be of 5 lines, you see in the sample output to span them to 7 lines which is due to formatting in word. You should output in 5 lines on your output.

J 10991 - Region



From above figure, it is clear that **C1**, **C2** and **C3** circles are touching each other.
Consider,

C1 circle have **R1** radius.
C2 circle have **R2** radius.
C3 circle have **R3** radius.

Write a program that will calculate the area of shaded region **G**

Input

The first line will contain an integer **k (1 ≤ k ≤ 1000)** which is the number of cases to solve. Each of the following k Lines will contain three floating point number **R1 (1 ≤ R1 ≤ 1000)**, **R2 (1 ≤ R2 ≤ 1000)** and **R3 (1 ≤ R3 ≤ 1000)**.

Output

For each line of input, generate one line of output containing the area of **G** rounded to six decimal digits after the decimal point. Floating-point errors will be ignored by special judge program.

Sample Input	Output for Sample Input
2 5.70 1.00 7.89 478.61 759.84 28.36	1.224323 2361.005761

K 10986 - Sending email

Time limit: 3.000 seconds

"A new internet watchdog is creating a stir in Springfield. Mr. X, if that is his real name, has come up with a sensational scoop."

Kent Brockman

There are **n** SMTP servers connected by network cables. Each of the **m** cables connects two computers and has a certain latency measured in milliseconds required to send an email message. What is the shortest time required to send a message from server **S** to server **T** along a sequence of cables? Assume that there is no delay incurred at any of the servers.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing **n** (2<=**n**<20000), **m** (0<=**m**<50000), **S** (0<=**S**<**n**) and **T** (0<=**T**<**n**). **S**!=**T**. The next **m** lines will each contain 3 integers: 2 different servers (in the range [0, **n**-1]) that are connected by a bidirectional cable and the latency, **w**, along this cable (0<=**w**<=10000).

Output

For each test case, output the line "Case #**x**:" followed by the number of milliseconds required to send a message from **S** to **T**. Print "unreachable" if there is no route from **S** to **T**.

Sample Input	Sample Output
3 2 1 0 1 0 1 100 3 3 2 0 0 1 100 0 2 200 1 2 50 2 0 0 1	Case #1: 100 Case #2: 150 Case #3: unreachable

L 10600 - ACM Contest and Blackout

In order to prepare the "The First National ACM School Contest"(in 20??) the major of the city decided to provide all the schools with a reliable source of power. (The major is really afraid of blackouts☺). So, in order to do that, power station "Future" and one school (doesn't matter which one) must be connected; in addition, some schools must be connected as well.

You may assume that a school has a reliable source of power if it's connected directly to "Future", or to any other school that has a reliable source of power. You are given the cost of connection between some schools. The major has decided to pick out two the cheapest connection plans – the cost of the connection is equal to the sum of the connections between the schools. Your task is to help the major – find the cost of the two cheapest connection plans.

Input

The Input starts with the number of test cases, T ($1 \le T \le 15$) on a line. Then T test cases follow. The first line of every test case contains two numbers, which are separated by a space, N ($3 \le N \le 100$) the number of schools in the city, and M the number of possible connections among them. Next M lines contain three numbers A_i, B_i, C_i , where C_i is the cost of the connection ($1 \le C_i \le 300$) between schools A_i and B_i. The schools are numbered with integers in the range 1 to N.

Output

For every test case print only one line of output. This line should contain two numbers separated by a single space - the cost of two the cheapest connection plans. Let S_1 be the cheapest cost and S_2 the next cheapest cost. It's important, that $S_1 = S_2$ if and only if there are two cheapest plans, otherwise $S_1 \le S_2$. You can assume that it is always possible to find the costs S_1 and S_2..

Sample Input	Sample Output
2 5 8 1 3 75 3 4 51 2 4 19 3 2 95 2 5 42 5 4 31 1 2 9 3 5 66 9 14 1 2 4 1 8 8 2 8 11 3 2 8 8 9 7 8 7 1 7 9 6 9 3 2 3 4 7 3 6 4 7 6 2 4 6 14 4 5 9 5 6 10	110 121 37 37