ACM Programming Contest

December 16, 2000

Rules:

1. There are *eight* problems for each team to be completed in *five* hours.
2. All problems require you to read test data from the *standard input* and write the results to the *standard output*.
3. The name of the file containing the sample input given for each problem. If you test your program using PC^2, it will automatically redirect input from the sample input file to your program.
4. Output must correspond *exactly* to the provided sample output, including (mis)spellings and spacing. Multiple spaces will not be used in any of the judges? output, except where explicitly stated.
5. Programming style is not considered in this contest. You are free to code *in whatever style you prefer*. Documentation is not required.
6. Contestant teams *must* submit their program through PC^2 software.
7. Contestants may use *books* and *manuals*. Nothing else is allowed.
8. *Judges? decisions are final.*

Tip: *Try to solve the easiest problem first.*

Problem A

Boundary Walls

Sample Input: boundary.in

A cooperative society has purchased some land for distribution amongst its members. The land is divided into plots so that every member can get a plot. It is planned to mark each plot by a two meter high boundary wall. The boundary of each plot is shaped like a polygon (not necessarily a convex one). (The figure below shows a land with five plots.) It is divided into line segments such that any two line segments either intersect at an end point or do not intersect at all (there may be co-linear segments present). A builder has been hired to construct the boundary walls. He has put different groups of workers to work on different segments of the boundary wall (a segment of boundary wall corresponds to a line segment in the boundary of a plot). All the groups, at the end of the day, submit their report to the contractor indicating whether their work is finished or not. Write a program to help identify the builder the number of plots for which the work has been completed.

Input

The first line of the input gives the number of test cases T.

For each test case, the first line contains an integer N (between 3 and 50) giving the number of line segments in the land. The j-th of the next N lines contains five integers *a*, *b*, *c*, *d*, and *e* where (*a*, *b*) and (*c*, *d*) give the coordinates of the corner point of the j-th line segment and the number *e* is 1 if that boundary wall segment is complete, 0 otherwise.

Output

For each test case, there is one line in the output giving the number of plots on which work is completed.

Sample Input

2

5

0 0 0 1 0

0 0 1 0 1

0 1 1 1 1

1 0 1 1 1

0 0 1 1 1

7

0 0 2 0 1

0 0 0 1 1

2 0 2 2 1

2 2 1 2 1

1 2 0 1 1

1 2 1 1 1

0 1 1 1 1

Sample Output

1

2

PROBLEM B

Ski Slopes

Sample Input: ski.in

A skier wants to ski down from the top of a mountain to its base. There are several possible routes, using different slopes enroute, and passing through some flat areas. The effort expended in skiing down a slope depends upon the length of the slope and the speed of skiing. For each slope, there is a maximum advisable speed. The skier wants to use a route that minimizes the average effort spent per unit distance traveled (i.e., the total effort expended divided by the total distance traveled).

You are given the map of the mountain slopes. That is, the flat areas and the slopes connecting these areas are given. Note that on a slope, one can only ski downwards. For each slope, you are also given the length of the slope and the maximum advisable speed for it. The effort expanded in skiing down a particular slope is given by the following formula.

e = *d**(70 - *s*) if *s* £ 60, and *e* = *d**(*s* - 50) if *s* > 60

where *e* is the effort required, *d* is the distance traveled, and *s* is the speed of travel.

You have to determine the minimum *average* effort per unit distance that the skier has to expend in order to reach the mountain base, while staying within the maximum advisable speed at every slope.

Input

The input may have multiple test cases. The first line of input gives the number of test cases T. For each test case, the first line of input gives the number of flats, *N* (*N* £ 100), and the number of slopes, *R* (*R* £ 10000), connecting them respectively. The flats are assumed to be numbered from 1 to *N*. The flats at the top and the base of the mountain are assumed to be numbered 1 and *N* respectively. Each of the next *R* lines describes a slope by giving: the numbers of the flats at the top and the bottom of the slope, the maximum advisable speed for the slope, and the length of the slope respectively.

Output

For each test case, output a single number (to an accuracy of 2 decimal places) that gives the minimum average effort per unit distance that needs to be expended to ski down from the mountain top to the base. The output for each test case should be on a separate line.

Sample Input

2

4 5

1 4

1 4 30 60

1 2 50 40

1 3 60 20

2 4 60 50

3 4 50 50

3 3

1 3

1 2 50 40

1 3 40 20

2 3 20 30

Sample Output

14.44

30.00

PROBLEM C

The Monkey Dance

Sample Input: dance.in

The director of Hind Circus has decided to add a new performance, the monkey dance, to his show. The monkey dance is danced simultaneously by N monkeys. There are N circles drawn on the ground. In the beginning, each monkey sits on a different circle. There are N arrows drawn from circle to circle in such a way that in each circle, exactly one arrow starts and exactly one ends. No arrow can both begin and end at the same circle. When the show begins, at each whistle of the ringmaster, all the monkeys simultaneously jump from their respective circles to other circles following the arrows from their respective current circles. This is one step of the dance. The dance ends when *all* the monkeys have returned to the circles where they initially started. The director wishes the dance to last as many steps as possible. This can be achieved by drawing the arrows intelligently. Given N, you have to write a program to determine the maximum possible number of steps such that there is a way of drawing the arrows to make the dance last these many steps.

Input

The input may have multiple cases. Each case consists of just the value of N (N £ 40) on a separate line. The input ends with a 0 for the value of N.

Output

For each case, simply output the maximum possible number of steps. Each output should be on a separate line.

Sample Input

5

8

0

Sample Output

6

15

Problem D

Error Correcting Polynomials

Sample Input: polynomial.in

Error correction is extremely useful in recovering from corrupted data. The abstract setting for this problem is: given a set of n data elements such that at least t of the data elements are uncorrupted, recover the complete uncorrupted data from it. To be able to do such a correction, data elements need to be of special form. Such forms are called *Error-correcting codes*. An error-correcting code that is widely used is as follows: the data elements are evaluations of a univariate polynomial of degree d modulo a prime number p. The error-correction algorithm for this code was very complex until Madhu Sudan discovered a nice way of doing this. His algorithm is: given a set of n points y1, ?, yn (each value is between 0 and p-1, and also $n < p$) with at least t of the ys correct (this means that for the correct yj, yj = P(j) for some unknown degree d polynomial modulo p) do the following:

1. Construct a *non-zero* polynomial in two variables $Q(x, y)$ with the degree of y being e and the degree of x being f and $(e+1)*(f+1) > n$ and $e * d + f < t$ for which Q(j, yj) = 0 for every j between 1 and n. The exact values of e and f are determined using t, d, and n.

2. Factor the polynomial Q into irreducible factors.

3. Consider all the factors of the form $(y - g(x))$ such that degree of g is d and identify one for which yj = g(j) for at least t values.

This identified polynomial, g, is the desired one. Once the polynomial is identified, all the correct values of y can be recovered by computing g(j) for every j.

Madhu Sudan has written programs for factoring the two variable polynomial and for identifying the right polynomial as well as computed the right values of e and f. He now just needs a program that constructs the two variable polynomial Q. Please help him in writing this program.

Input

The number of test cases T is written in the first line of the input.

For each test case, the first line contains the value of n, p, e, and f (all these are between 1 and 100 satisfying the constraints mentioned above; in particular, p is always a prime number) in that order. The j-th of the next n lines contains the value yj.

Output

For each test case, output all the coefficients of the polynomial Q in the following order: first output all the coefficients of the terms in which the degree of y is zero and order these coefficients by increasing degree of x; then output all the coefficients of the terms in which degree of y is one ordering these in the same way as before, and so on. Each coefficient must be on a separate line. If there are more than one solutions, output any one.

The output of different test cases must be separated by a blank line.

Sample Input

2

3 7 1 1

6

4

5

4 11 1 2

2

3

0

0

Sample Output

0

4

3

1

1

4

1

2

6

0

Problem E

Spaceship Travel

Sample Input: space.in

A spaceship is to be launched from the Earth to visit a number of planets. The order of visiting the planets is fixed. The spaceship can travel from one planet to another via normal space or via *hyperspace*. Hyperspace travel reduces the traveled distance substantially, however, the fuel consumption during the hyperspace travel is much more. In fact, if the spaceship travels *d* light years thorough the hyperspace, then the fuel consumed is d^4 *units! For d light years traveled through normal space, the fuel consumed is just d units. Also, between two planets, the spaceship can either travel via hyperspace or normal space but not both. Design an algorithm that determines a way of traveling that minimizes the fuel consumption.*

Input

The first line of the input file contains a number T giving the number of test cases to follow.

The first line of each test case contains an integer N (between 1 and 50) giving the number of planets to be visited. The j-th of the next N lines correspond to the j-th planet to be visited and contains two integers *hd* and *nd* denoting respectively the hyperspace and normal space distance between the (j-1)-th planet and j-th planet (the 0-th planet is taken to be the Earth). The number *hd* is between 1 and 10 light years and the number *nd* is between 1 light year and 10 million light years.

Output

For each test case in the input, output a number giving the minimum units of fuel required to visit all the planets. Each test case output must be on a separate line.

Sample Input

2

3

3 1000

2 5000

4 8000

5

1 10000

2 3547

7 36782

4 2178

9 67428

Sample Output

2296

52507

Problem F

Maze

Sample Input: maze.in

One day, looking at old documents inherited from his grandfather, Ramesh discovers a map. Reading the accompanying document, he realizes that the map is of a complex maze of chambers and passages inside a mountain. Chambers in this maze have doors opening to passages leading to other chambers. All the doors have one peculiarity: they have handles only on one side, and so can be opened either from the chamber or from the passage but not both. The passages leading from any chamber have been designed in such a way that either there is only one passage leading out or there is a passage such that upon following it one can *never come back* to the chamber. Also, there is at most one passage between any two chambers. On the ceiling of each chamber, a number is written. There are two special chambers: the entry chamber and the exit chamber. The entry chamber can be entered via a door from outside the mountain and the exit chamber has a door that leads to a secret chamber. The exit door opens only upon reciting a magic sequence of numbers. It is given that this sequence is the only sequence of numbers satisfying the following properties:

- The sequence can be constructed by noting the numbers written in chambers, in that order, on a path from entry chamber to exit chamber.
- The numbers in the sequence are all distinct.
- If any chamber has a number from this sequence written, then there is a path from the chamber to the exit chamber such that all the chambers in this path contain numbers *only* from this sequence.

Design an algorithm to help Ramesh reach the secret chamber.

Input

The first line of the input consists of an integer T giving the number of test cases to follow.

The first line of each test case contains two integers N (between 1 and 100) and P (between 1 and 1000) giving the number of chambers and the number of passages in the maze respectively. The next line contains N numbers: the j-th number is written in the j-th chamber. Each of the next P lines contain four numbers a, b, c, d with a, b between 1 and N and c, d having binary value, giving that there is a passage between chamber number a and chamber number b such that the door to the passage from chamber a opens from inside the chamber iff $c = 1$ and the door to the passage from chamber b opens from inside the chamber iff $d = 1$. The last line of the test case contains two numbers s and t with s being the entry chamber number and t being the exit chamber number.

Output

For each test case in the input, there is one output line that contains the magic sequence with numbers separated by blanks.

Sample Input

2

6 6

9 10 11 12 13 14

1 2 1 0

2 3 1 0

3 4 1 0

4 5 1 0

4 6 1 0

4 2 1 0

1 5

10 13

22 81 84 84 72 60 63 62 99 54

1 2 1 0

1 4 1 1

2 4 1 0

2 5 1 0

5 3 0 1

6 3 0 1

4 7 1 0

8 4 0 1

5 2 1 0

5 8 1 0

9 5 0 1

9 6 0 1

10 6 0 1

1 9

Sample Output

9 10 11 12 13

22 81 72 99

Problem G

Terrain Traversal

Sample Input: terrain.in

A battalion of tanks is to be sent to a border post from the base. There are no obstructions between the base and the border post and so the tanks can travel in any direction. However, the terrain between the base and the border post is of two kind: first comes the muddy terrain and then the desert. The maximum velocity of a tank in each of these terrains is different. It is assumed that a straight line separates the two terrains. Design an algorithm that finds out the fastest route to the border post.

Input

There are several test cases. The first line of the input contains the number of test cases T.

For each test case, the first line has two numbers u and v with u being the maximum velocity (in kilometers per hour) of a tank in the desert and v the maximum velocity in the muddy terrain. The next line contains the coordinates of the border post assuming the dividing line between the terrains to be the x-axis (the y-coordinate of the border post is always positive). The third line contains the coordinates of the base camp (the y-coordinate of the base camp is always negative). The unit of the coordinates is kilometers.

Output

For each test case, there is one output line containing two numbers: the first number denotes the time taken in seconds (rounded off to the closest second) by a tank to reach the border post by the fastest route. The second number denotes the x-coordinate (in kilometers, with an accuracy of three decimal digits) of the point where the tank crosses over to the desert.

Sample Input

2

1 1

5 3

5 -2

3 5

20 10

8 -9

Sample Output

18000 5.000

21590 15.688

Problem H

Acid Containers

Sample Input: acid.in

Company ACIDIC manufactures acid. The acid is filled into containers and sent out to different places. Sometimes, the containers start leaking. To handle such situations, the company has a room with several container holders. These holders are arranged in a rectangle grid. Each of these can hold one leaky container and absorb the acid leakage. After all the acid has leaked out, the containers are taken out and the leak point(s) are fixed. One day, the leaky containers were put in some of the holders. After a while, some more leaky containers were detected and brought to the room. The supervisor handling this then realized that all the containers that are present in the room are leaking very heavily. So heavily that they have corroded their holders and in a short time they would also corrode all the holders in the room in the directions of the leak. The containers are leaking either in the North-South direction or in the East-West direction. (The figure above shows a room with a 4-by-6 grid of holders having four leaky containers. The dashed arrows represent the direction of leakage.) The supervisor tried to remove containers from their holders but could not as they were stuck to the holders. However, he could rotate the containers so that the direction of leakage could be changed from North-South to East-West or vice-versa. The supervisor suspects that the current lot of leaky containers may also start to leak heavily. Help the supervisor find the rotations of existing containers and placement of current lot of containers so that the number of corroded holders is minimized.

Input

There are several test cases. The first line of the input contains the number of test cases T.

For each test case, the first line has four numbers R, C, N and M. Numbers R and C (between 10 and 100) give the number of rows and columns in the grid. Number N (between 1 and 20) gives the number of leaky containers in the room and number M (between 1 and 20) gives the number of leaky containers in the current lot. Each of the next N lines gives the placement and the leakage direction of an existing container. These are given as three numbers *a*, *b*, and *c* where *a* (between 1 and R) is the row number (counting from top), *b* (between 1 and C) is the column number (counting from left), and *c* is 1 if the leakage direction is North-South, 0 otherwise.

Output

For each test case, the output should be a number giving the smallest number of holders that will get corroded after rotating the existing containers and placing the containers in the current lot (assuming the all the current lot containers will also leak heavily). The output for different test cases should be on different lines.

Sample Input

2

4 6 4 4

1 2 0

2 4 0

3 2 1

3 5 1

50 50 5 10

1 35 1

17 44 0

17 46 1

42 35 1

42 46 0

Sample Output

12

148