Nordic Collegiate Programming Contest NCPC 2006

September 30, 2006

The Problemset

A	Shoot-out
B	Tour Guide
C	Nasty Hacks
D	Jezzball
E	Card Trick
F	Traveling Salesman
G	Whac-a-Mole
H	Random Walking
I	Honeycomb Walk

Problem A

Shoot-out

This is back in the Wild West where everybody is fighting everybody. In particular, there are n cowboys, each with a revolver. These are rather civilized cowboys, so they have decided to take turns firing their guns until only one is left standing. Each of them has a given probability of hitting his target, and they all know each other's probability. Furthermore, they are geniuses and always know which person to aim at in order to maximize their winning chance, so they are indeed peculiar cowboys. If there are several equally good targets, one of those will be chosen at random. Note that a cowboy's code of ethics forces him to do his best at killing one of his opponents, even if intentionally missing would have increased his odds (yes, this can happen!)

Input specifications

On the first line of the input is a single positive integer t, telling the number of test cases to follow. Each case consists of one line with an integer $2 \leq n \leq 13$ giving the number of cowboys, followed by n positive integers giving hit percentages for the cowboys in the order of their turns.

Output specifications

For each test case, output one line with the percent probabilities for each of them surviving, in the same order as the input. The numbers should be separated by a space and be correctly rounded to two decimal places.

Sample input

```
5
2 1 100
3 100 99 98
3 50 99 100
3 50 99 99
3 50 99 98
```

Output for sample input

```
1.00 99.00
2.00 0.00 98.00
25.38 74.37 0.25
25.38 49.50 25.12
25.63 24.63 49.74
```

Problem B

Tour Guide

You are working as a guide on a tour bus for retired people, and today you have taken your regular Nordic seniors to The Gate of Heavenly Peace. You let them have a lunch break where they could do whatever they like. Now you have to get them back to the bus, but they are all walking in random directions. You try to intersect them, and send them straight back to the bus. Minimize the time before the last person is in the bus. You will always be able to run faster than any of the tour guests, and they walk with constant speed, no matter what you tell them. The seniors walk in straight lines, and the only way of changing their direction is to give them promises of camphor candy. A senior will neither stop at nor enter the bus before given such a promise.

Input specifications

A number of test cases consisting of: A line with an integer $1 \leq n \leq 8$, the number of people on the tour. A line with an floating point number $1 < v \leq 100$, your maximum speed (you start in the bus at the origin). Then follow n lines, each containing four floating point numbers x_i y_i v_i a_i, the starting coordinates ($-10^6 \leq x_i, y_i \leq 10^6$), speed ($1 \leq v_i < 100$) and direction ($0 \leq a_i < 2\pi$) of each of the tour guests.

The input is terminated by a case with $n = 0$, which should not be processed. All floating point numbers in the input will be written in standard decimal notation, and have no more than 10 digits.

Output specifications

For each test case, print a line with the time it takes before everybody is back in the bus (the origin). Round the answer to the nearest integer. The answer will never be larger than 10^6.

Sample input

```
1
50.0
125.0 175.0 25.0 1.96
3
100.0
40.0 25.0 20.0 5.95
-185.0 195.0 6.0 2.35
30.0 -80.0 23.0 2.76
0
```

Output for sample input

```
20
51
```

Problem C

Nasty Hacks

You are the CEO of Nasty Hacks Inc., a company that creates small pieces of malicious software which teenagers may use to fool their friends. The company has just finished their first product and it is time to sell it. You want to make as much money as possible and consider advertising in order to increase sales. You get an analyst to predict the expected revenue, both with and without advertising. You now want to make a decision as to whether you should advertise or not, given the expected revenues.

Input specifications

The input consists of n cases, and the first line consists of one positive integer giving n. The next n lines each contain 3 integers, r, e and c. The first, r, is the expected revenue if you do not advertise, the second, e, is the expected revenue if you do advertise, and the third, c, is the cost of advertising. You can assume that the input will follow these restrictions: $-10^6 \leq r, e \leq 10^6$ and $0 \leq c \leq 10^6$.

Output specifications

Output one line for each test case: "advertise", "do not advertise" or "does not matter", presenting whether it is most profitable to advertise or not, or whether it does not make any difference.

Sample input

```
3
0 100 70
100 130 30
-100 -70 40
```

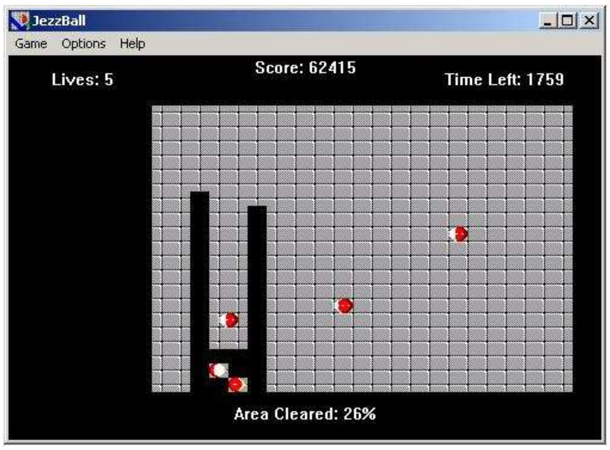
Output for sample input

```

advertise
does not matter
do not advertise
```

Problem D

Jezzball



"JezzBall is a computer game in which red-and-white 'atoms' bounce about a rectangular field of play. The player advances to later levels (with correspondingly higher numbers of atoms and lives) by containing the atoms in progressively smaller spaces, until at least 75% of the area is blocked off." (wikipedia.org)

The picture to the right is a screenshot from the original game, where the player has already covered some space (the black part). In this problem we will consider a slightly different, non-discrete, version of the game. That is, while the length unit is still pixels, you should treat them as non-discrete in the sense that all objects can be at non-integer coordinates and all movements are continuous.

The size of the playing field will be 1024×768 pixels. The atoms that bounce around will be infinitely thin (and not round balls like in the screenshot). The atoms will move at a constant speed and only change direction when hitting the edge of the playing field (x-coordinate 0 and 1024 or y-coordinate 0 and 768), where they bounce without loss of energy. The atoms do not hit each other.

The player can divide the playing field in two by shooting a horizontal or vertical ray from (in this problem) a fixed point on the playing field. The ray will then extend in both directions simultaneously (up and down for vertical rays, or left and right for horizontal rays) at a uniform speed (in this problem always 200 pixels per second). The rays will also be infinitely thin. If no atom touches any part of the ray while it's still being extended, the field has sucessfully been divided. Otherwise the player loses a life.

If an atom touches the endpoint of an extending edge, this will not be counted as a hit. Also, if an atom hits the ray at the same instant it has finished extending, this will also not count as a hit. Write a program that determines the minimum time the player must wait before he can start extending a ray so that an atom will not hit it before the ray has been completed.

Input specifications

Each test case starts with a line containing a single integer n, the number of atoms ($1 \leq n \leq 10$). Then follows a line containing two integers, x and y, the position where the two ray ends will start extending from ($0 < x < 1024$, $0 < y < 768$). Then n lines follow, each containing four integers, x, y, v_x and v_y describing the initial position and speed of an atom ($0 < x < 1024$, $0 < y < 768$, $1 \le |v_x| \le 200$, $1 \le |v_y| \le 200$). The speed of the atom in the x direction is given by v_x, and the speed in the y direction is given by v_y. All positions in each input will be distinct. The input is terminated by a case where $n = 0$, which should not be processed. There will be at most 25 test cases.

Output specifications

For each test case, output the minimum time (with exactly 5 decimal digits) until the player can extend either a horizontal or vertical ray without an atom colliding with it while it is being drawn. The input will be constructed so that the first time this occurs will be during an open interval at least 10^{-5} seconds long. If no such interval is found during the first 10000 seconds, output "Never" (without quotes).

Sample input

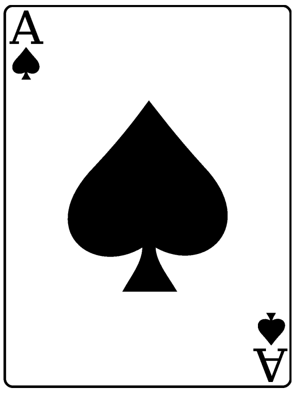
```
3
700 420
360 290 170 44
900 150 -53 20
890 100 130 -100
4
10 10
1 1 192 144
513 385 192 144
1023 767 -192 -144
511 383 -192 -144
0
```

Output for sample input

```
2.80094
Never
```

Problem E

Card Trick



The magician shuffles a small pack of cards, holds it face down and performs the following procedure:

1. The top card is moved to the bottom of the pack. The new top card is dealt face up onto the table. It is the Ace of Spades.
2. Two cards are moved one at a time from the top to the bottom. The next card is dealt face up onto the table. It is the Two of Spades.
3. Three cards are moved one at a time …
4. This goes on until the *n*th and last card turns out to be the *n* of Spades.

This impressive trick works if the magician knows how to arrange the cards beforehand (and knows how to give a false shuffle). Your program has to determine the initial order of the cards for a given number of cards, $1 \leq n \leq 13$.

Input specifications

On the first line of the input is a single positive integer, telling the number of test cases to follow. Each case consists of one line containing the integer n.

Output specifications

For each test case, output a line with the correct permutation of the values 1 to n, space separated. The first number showing the top card of the pack, etc …

Sample input

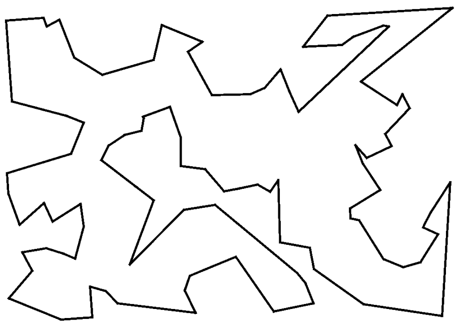
```
2
4
5
```

Output for sample input

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

Problem F

Traveling Salesman



Long before the days of international trade treaties, a salesman would need to pay taxes at every border crossed. So your task is to find the minimum number of borders that need to be crossed when traveling between two countries. We model the surface of Earth as a set of polygons in three dimensions forming a closed convex 3D shape, where each polygon corresponds to one country. You are not allowed to cross at points where more than two countries meet.

Input specifications

Each test case consists of a line containing c, the number of countries ($4 \leq c \leq 6000$), followed by c lines containing the integers $n\ x_1\ y_1\ z_1\ \ldots\ x_n\ y_n\ z_n$, describing (in order) the n corners of a closed polygon ($3 \leq n \leq 20$). Then follows a line with one integer m ($0 < m \leq 50$), and then m lines with queries $c_a\ c_b$, where c_a and c_b are country numbers (starting with 1). No point will be on the line between two connected points, and $-10^6 \leq x, y, z \leq 10^6$ for all points. No two non-adjacent edges of a country share a common point. The input is terminated by a case where $c = 0$, which should not be processed.

Output specifications

For each query, output the number of borders you must cross to go from c_a to c_b.

Sample input

```
6
4 0 0 0 0 0 1 0 1 1 0 1 0
4 1 0 0 1 0 1 1 1 1 1 1 0
4 0 0 0 1 0 0 1 0 1 0 0 1
4 0 1 0 1 1 0 1 1 1 0 1 1
4 0 0 0 0 1 0 1 1 0 1 0 0
4 0 0 1 0 1 1 1 1 1 1 0 1
2
1 2
1 3
0
```

Output for sample input

```
2
1
```

Problem G

Whac-a-Mole

While visiting a traveling fun fair you suddenly have an urge to break the high score in the Whac-a-Mole game. The goal of the Whac-a-Mole game is to... well... whack moles. With a hammer. To make the job easier you have first consulted the fortune teller and now you know the exact appearance patterns of the moles.

The moles appear out of holes occupying the n^2 integer points (x, y) satisfying $0 \leq x, y < n$ in a two-dimensional coordinate system. At each time step, some moles will appear and then disappear again before the next time step. After the moles appear but before they disappear, you are able to move your hammer in a straight line to any position (x_2, y_2) that is at distance at most d from your current position (x_1, y_1). For simplicity, we assume that you can only move your hammer to a point having integer coordinates. A mole is whacked if the center of the hole it appears out of is located on the line between (x_1, y_1) and (x_2, y_2) (including the two endpoints). Every mole whacked earns you a point. When the game starts, before the first time step, you are able to place your hammer anywhere you see fit.

Input specifications

The input consists of several test cases. Each test case starts with a line containing three integers n, d and m, where n and d are as described above, and m is the total number of moles that will appear ($1 \leq n \leq 20$, $1 \leq d \leq 5$, and $1 \leq m \leq 1000$). Then follow m lines, each containing three integers x, y and t giving the position and time of the appearance of a mole ($0 \leq x, y < n$ and $1 \leq t \leq 10$). No two moles will appear at the same place at the same time.

The input is ended with a test case where $n = d = m = 0$. This case should not be processed.

Output specifications

For each test case output a single line containing a single integer, the maximum possible score achievable.

Sample input

```
4 2 6
0 0 1
3 1 3
0 1 2
0 2 2
1 0 2
2 0 2
5 4 3
0 0 1
1 2 1
2 4 1
0 0 0
```

Output for sample input

```
4
2
```

Problem H

Random Walking

The Army of Coin-tossing Monkeys (ACM) is in the business of producing randomness. Good random numbers are important for many applications, such as cryptography, online gambling, randomized algorithms and panic attempts at solutions in the last few seconds of programming competitions.

Recently, one of the best monkeys has had to retire. However, before he left, he invented a new, cheaper way to generate randomness compared to directly using the randomness generated by coin-tossing monkeys. The method starts by taking an undirected graph with 2^n nodes labelled $0, 1, \ldots, 2^n - 1$. To generate k random n-bit numbers, they will let the monkeys toss n coins to decide where on the graph to start. This node number is the first number output. The monkeys will then pick a random edge from this node, and jump to the node that this edge connects to. This new node will be the second random number output. They will then select a random edge from this node (possibly back to the node they arrived from in the last step), follow it and output the number of the node they landed on. This walk will continue until k numbers have been output.

During experiments, the ACM has noticed that different graphs give different output distributions, some of them not very random. So, they have asked for your help testing the graphs to see if the randomness is of good enough quality to sell.

They consider a graph good if, for each of the n bits in each of the k numbers generated, the probability that this bit is output as 1 is greater than 25% and smaller than 75%.

Input specifications

The input will consist of several data sets. Each set will start with a line consisting of three numbers k, n, e separated by single spaces, where k is the number of n-bit numbers to be generated and e is the number of edges in the graph ($1 \leq k \leq 100$, $1 \leq n \leq 10$ and $1 \leq e \leq 2000$). The next e lines will consist of two space-separated integers v_1, v_2 where $0 \leq v_1, v_2 < 2^n$ and $v_1 \neq v_2$. Edges are undirected and each node is guaranteed to have at least one edge. There may be multiple edges between the same pair of nodes.

The last test case will be followed by a line with $k = n = e = 0$, which should not be processed.

Output specifications

For each input case, output a single line consisting of the word `Yes` if the graph is good, and `No` otherwise.

Sample input

```
10 2 3
0 3
1 3
2 3
5 2 4
0 1
0 3
1 2
2 3
0 0 0
```

Output for sample input

```
No
Yes
```

Problem I

Honeycomb Walk

A bee larva living in a hexagonal cell of a large honeycomb decides to creep for a walk. In each "step" the larva may move into any of the six adjacent cells and after n steps, it is to end up in its original cell.

Your program has to compute, for a given n, the number of different such larva walks.

Input specifications

The first line contains an integer giving the number of test cases to follow. Each case consists of one line containing an integer n, where $1 \leq n \leq 14$.

Output specifications

For each test case, output one line containing the number of walks. Under the assumption $1 \leq n \leq 14$, the answer will be less than 2^{31}.

Sample input

```
2
2
4
```

Output for sample input

```
6
90
```