7th South African Regional
ACM Collegiate Programming Competition

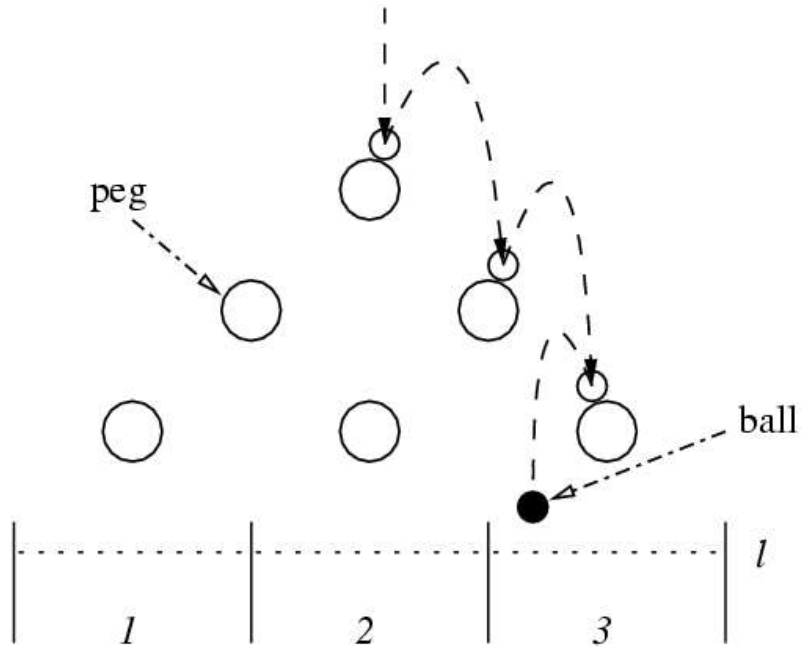
Sponsored by IBM

Problem D – Blue balloon
Feeling lucky

You have just made a bet with a friend: he believes in the inherent randomness of the *Pachinko* game, while you believe that it is governed by simple physics, and is therefore predictable. To prove your bet, you have to set up an experiment, and predict beforehand where the ball will end up.
But first, the more fundamental question: What on earth is a *Pachinko* game?

The diagram below illustrates the set-up: The game consists of a board, mounted vertically, with a number of cylindrical pegs fitted to it. The diagram illustrates the front view, *i.e.* the cylinders are sticking out of the page (and are thus parallel to the surface of the earth). Below the pegs, a number of "bins" have been painted on the board, along with the "finish line", *l*.

The game starts when a small metal ball is released somewhere above the topmost peg. The ball then falls under the influence of gravity, bouncing off the pegs, until it crosses the line *l*. At the instant that the center of the ball falls exactly on the line *l*, the bin which the ball falls inside of is recorded as the "winning bin".



Some people believe that the distribution of the balls (amongst the bins) can be influenced using psychic powers. You have wrapped some tinfoil around your friend's head, just in case.

Your task is to write a simulation program to calculate in which bin the ball will end up, given that you know the exact starting position of the ball. You assumed the following physical model of the game:

1. The ball falls under the influence of gravity, and experiences linear drag.
2. The ball starts falling from the initial position with an initial velocity of [0,0].
3. The collisions between the ball and the peg are elastic, *i.e.* no energy is lost.
4. The ball does not experience any additional friction when it makes contact with the pegs, therefore it does not rotate during collisions, it just bounces.
5. The initial direction in which the ball bounces off the peg is calculated by reflecting the incoming direction around the normal vector of the peg at the point of contact.

The motion of a body in free fall, experiencing only linear drag, can be represented using the equations

$$\mathbf{P}(t) = \mathbf{C}_v \frac{m}{k} e^{k \cdot t/m} - \mathbf{G}\frac{m \cdot g \cdot t}{k} + \mathbf{C}_p$$

and

$$\mathbf{V}(t) = \mathbf{C}_v e^{k \cdot t/m} - \mathbf{G}\frac{m \cdot g}{k}$$

where

P(t)	=	a vector indicating the position of the object at time *t*
V(t)	=	a vector indicating the velocity of the object at time *t*
t	=	time in seconds, *t*=0 when ball is released
$\boldsymbol{C}_v$	=	$\mathbf{V}(0) + \mathbf{G}\frac{m \cdot g}{k}$
V(0)	=	[initial horizontal velocity, initial vertical velocity]
G	=	[0, -1]. This is the direction of gravity
m	=	mass of object, in kg
g	=	9.8 ms^{-2}
k	=	viscosity coefficient = -1.0
$\boldsymbol{C}_p$	=	$\mathbf{P}(0) - \mathbf{C}_v \frac{m}{k}$
P(0)	=	[initial horizontal position, initial vertical position]

You may further assume the following values for the remaining constants:

Mass of ball, *m*	=	1.25
Radius of ball	=	0.5
Radii of pegs	=	1.0

You may notice that your simulation (and thus your physical experiment) has rather large dimensions. Just in case the tinfoil fails.

Input

Your input is a description of the configuration of the pegs and the bins, followed by a number of initial positions for the ball. The input thus assumes the following format:

```
<num. of pegs, N>
<peg_1 x> <peg_1 y>
<peg_2 x> <peg_2 y>
...       ...
<peg_N x> <peg_N y>
```

```
<num. of bins> <leftmost_bin x> <finish line y> <bin width>

<ball_1 x> <ball_1 y>
<ball_2 x> <ball_2 y>
...        ...
-999
```

Note that an arbitrary number of ball coordinates may follow the last bin description, with the end of input indicated by a single value of -999. These initial positions represent sequential runs of the same peg configuration, but with different initial positions for the ball. There will only be one ball in the game at a time.

The bins are evenly spaced, with the first bin having an x-interval of
[<leftmost_bin x>, <leftmost_bin x> + <bin width>),
and the second at
[<leftmost_bin x> + <bin width>, <leftmost_bin x> + 2*<bin width>)
and so on.

The value <finish line y> is the *y* value of the finish line *l*, that is, the bin into which the ball has fallen must be determined when the ball has reached this *y* value.

Output

For each initial ball position in the input, you must print out the number of the bin in which the ball will land when it crosses the finish line.

Bins are numbered from 1 through the number specified in the input.

Important: The time of contact, *t*, in all collisions must be calculated accurate to a resolution of 10^{-10}.

Sample Input

```
3
 0 4
-4 0
 4 0
3  -8  -4  8
 0.5 6
-0.4 6
-999
```

Sample Output

```
2
2
```