3347 Dome of Tuxville

It has been many years since the MacroSolved Cybercomputer has taken control of earth and enslaved most of the humanity to power its operation through some biological pro cess with an enormous matrix of cells each containing a paralyzed human prison. Now, even the city of Tuxville, the only know remaining habitat where humans can walk freely, is constantly raided and bombarded by fleets of giant flying robotic bugs sent by MacroSolved.

The engineers in Tuxville has recently developed a new technology called shield projector for protecting the city. The projector is placed on the ground and is fed enough energy so that it creates a semi-sphere dome of radius R, within which no robotic bugs can enter. Of course the required energy is a strictly increasing function of R. The city council has requested the engineers to find the best spot to place the shield projector, so that it requires the minimal amount of energy while generating a dome of shield large enough to protect every building in the city. As a junior member of the engineering team, you are to compute the desired position (the x, y coordinates) for the projector given the list of positions and heights of all buildings in Tuxville.

Input

The input file consists of several test cases. Each test case begins with an integer N, $1 \leq N \leq 30$, the number of buildings in Tuxville. Each of the next N lines contains 3 real numbers x, y, and h which gives the position and the height of one building. The last test case is followed by a line consisting of a single '0'.

Every real number t in the input file has at most 3 digits after the decimal point and $-999.999 \leq t \leq 999.999$.

Note that there may be two or more buildings of equal or different heights occupying the same position, such as buildings 3 and 5 in the second test case. There may also be three or more buildings lying on the same straight line as exemplified by buildings 1, 2, and 4 in the second test case.

Output

Print the x, y coordinates of the desired position for each test case on one line. Round all numbers to 3 decimal places, printing the trailing 0's after the decimal place as necessary.

Sample Input

```
4
 2.0  1.0  0.2
 1.0 -2.0  0.3
-0.2  1.3  7.9
-2.0  1.0  0.1
5
 1.0  8.0  4.0
 8.0 -4.0  1.0
-4.0  1.0  8.0
 4.5  2.0  2.0
-4.0  1.0  3.0
0
```

Sample Output

```
-0.200    1.300
 0.000    0.000
```