Input: skroutz.in
Balloon Color: Dark Blue

Description

As your graduation approaches (or so you would like to believe!) you seek possible employment opportunities and as luck would have it you land a job with none other but Mr. Skroutz!

Mr. Skroutz is obsessive about his Internet bill and would like to reduce even further. While monitoring the stock market, a series of 32 bit integer numbers are send to his machine, but his keen eye observes that the numbers are not using all the 32 bits but only a small subset.

So, he hires you (without pay at first of course) to try and pack the numbers in as a small space as possible. For example, the following sequence of numbers that use only 7 bits:

Decimal	Binary
10	0001010
113	1110001
24	0011000
56	0111000

could be packed in a single 32 bit number as follows (leading zeros are added as only 28 bits are actually used):

Decimal	Binary
117848202	00000111000001100011100010001010

reducing the total space needed by a whooping 75% !

Input Format

The input will start with an integer *T* representing the number of test cases. Each test case starts with a line holding two numbers separated by whitespace: *N* and *B*, where *N* is how many positive integers follow and *B* how many bits are actually used in them. *N* can be between 1 and 100 and *B* can be between 1 and 32 inclusive.

Output Format

For each test case, print the list of (signed) integers produced by the packing process in a single line. The numbers should be separated by a single space.

Sample Input / Output

skroutz.in

```
3
4 7
10
113
24
56
12 4
1
2
3
4
5
6
7
8
9
10
11
12
1 1
1
```

OUTPUT

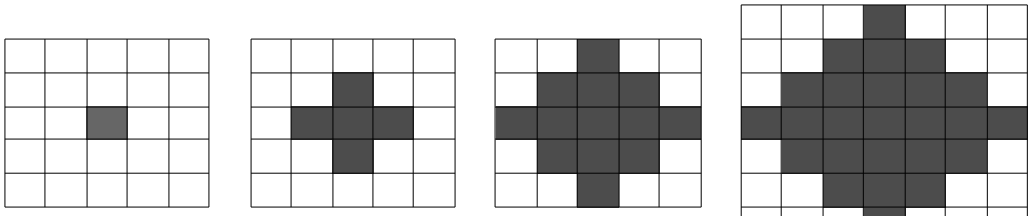
```
117848202
-2023406815 52137
1
```

Explosion!

Source file: explosion.(c|cpp|java)
Input: explosion.in
Balloon Color: Green

Description

Just how big can an explosion get? You are about to find out! It is assumed that the explosion starts from a single point in space modeled as a square and propagates to all four directions, one square per time unit. The progression is shown below:



Time : 0
Squares : 1

Time : 1
Squares : 5

Time : 2
Squares : 13

Time : 3
Squares : 25

Your task is to find the total space (in number of squares) that an explosion occupies after a specified about of time.

Input Format

The input will start with an integer ($1 \leq N \leq 100$) representing the number of test cases. N lines follow, one for each test case. Each line holds an integer ($0 \leq T \leq 1{,}000$) representing the time units at which the size of the explosion is to be found.

Output Format

For each test case print the corresponding explosion size in a separate line.

Sample Input / Output

explosion.in

```
2
1
3
```

OUTPUT

```
5
25
```

Matrix Madness

Source file: matrix.(c|cpp|java)
Input: matrix.in

Balloon Color: Pink

Description

Prof. G. is having nightmares. A matrix filled up by numbers in the fashion shown below has been haunting his sleep for days. He thinks he can sum them up easily but the method just escapes him!

1	**2**	**3**	**4**	**5**
2	**3**	**4**	**5**	**6**
3	**4**	**5**	**6**	**7**
4	**5**	**6**	**7**	**8**

Your task is to find the total sum of the numbers in an arbitrarily sized matrix.

Input Format

The input will start with an integer ($1 \leq N \leq 100$) representing the number of test cases. N lines follow, one for each test case. Each line holds the matrix rows R and columns C separated by a single space ($1 \leq R, C \leq 100$).

Output Format

For each test case print the corresponding sum in a separate line.

Sample Input / Output

matrix.in

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

OUTPUT

```
1
3
64
90
```

Josephus Returns

Source file: josephus.(c|cpp|java)
Input: josephus.in
Balloon Color: Yellow

Description

In the classical Josephus problem, *N* number of people grouped in a circle are killed one after the other until only one person remains. The killing starts at a fixed position *D*, then *D*-1 people are skipped and the next person is killed, and so on. The circle becomes smaller every time one person is killed. For example, if *N* is 5 and *D* is 3, the sequence of killing is as follows: 3, 1, 5, 2, and the person standing at position number 4 survives. In this problem, *D* is incremented by one after each killing. Your task, given *N* and *D*, is to find out which position will be the surviving one.

Input Format

The first line of the input file contains an integer *T*, which represents the number of test cases in the file. *T* lines follow with two integers *N* ($0 < N \leq 1{,}000$), and *D* ($0 < D \leq 1{,}000{,}000$) per line.

Output Format

For each test case print one line containing the surviving position.

Sample Input / Output

josephus.in

```
2
5 3
7 1
```

OUTPUT

```
1
5
```

Reflection

Source file: reflection.(c|cpp|java)
Input: reflection .in
Balloon Color: Light Blue

Description

In computer science, reflection is the process by which a computer program can observe and modify its own structure and behavior at runtime. But in this problem, we are only interested in geometrical reflection of a point across the x-axis, the y-axis, or the origin. You are given pairs of points and you are required to determine whether they are reflections of each other across the x-axis, the y-axis, or the origin.

Input Format

The first line of the input file contains an integer *T*, which represents the number of test cases in the file. *T* lines follow with two pairs of integers on each. The first pair of integers on a line represents the x and y coordinates of the first point respectively, and the second pair describes the second point similarly. The absolute value of any integer in the file is less than or equal to 10,000.

Output Format

For each test case print one line containing one of the following;

- "`The points are reflections across the x-axis.`", if they reflect across the x-axis,
- "`The points are reflections across the y-axis.`", if they reflect across the y-axis,
- "`The points are reflections across the origin.`", if they reflect across the origin,
- or "`The points are not reflections!`", if none of the previous cases applies.

Sample Input / Output

reflection.in

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

OUTPUT

```
The points are reflections across the x-axis.
The points are reflections across the y-axis.
The points are reflections across the origin.
The points are not reflections!
```

Elegance

Source file: elegance.(c|cpp|java)
Input: elegance .in
Balloon Color: Purple

Description

In many programming competitions, the method of logging the time of a correct submission for contestants is based on minutes. It means that the system prints out the number of minutes it took the contestant to solve a question since the beginning of the contest. The process seems easy enough for us –programming folks– to understand and calculate, but for other spectators it seems like hard work.

Since we don't want to frighten anyone who passes by the competition hall and sees the scoreboard, we have decided to build an elegant solution for time logging. Instead of just using the minutes only, we shall include also hours in the system. And to make it more and more elegant, we need to have it in a nice format that would be easy to read.

An example on that, 90 minutes should be converted to "`1 hour, 30 minutes`". Now it is easier to read. Note that you have to add 's' for plural number of minutes or hours. Also, if you get zero minutes or zero hours, you should remove that entire portion.

Input Format

The input starts with a number T that represents the number of test cases in the file. Each test case contains one number n ($1 \le n \le 100{,}000$) which is the number of minutes.

Output Format

The output format should follow the sample output below which covers all the possible formats.

Sample Input / Output

elegance.in

```
4
90
60
121
30
```

OUTPUT

```
1 hour, 30 minutes
1 hour
2 hours, 1 minute
30 minutes
```

Uncle Rashed

Source file: farm.(cpp|java)
Input: farm.in
Balloon Color: Blue

Description

Being the best farmer in the country, the government decided to give Uncle Rashed an area in the desert as a reward. However, there was a small problem. The area was a circular area, and Uncle Rashed is used to rectangular areas. In fact, he has a unique way of farming that made him the best farmer, and this way requires a rectangular area. Therefore, Uncle Rashed decided to farm only a rectangular area of the whole given area while keeping the remaining part as a wasteland (he might use it for any other purpose). However, it was very hard for him to get the largest possible rectangle. After hours of trying all possible scenarios (He was using a brute force algorithm) he came up with the best possible way (at least, this is what he claims). Being his friend, Uncle Rashed asked from you to write a code that calculates the area of the largest rectangle that could be formed inside a given circle. He wanted to know this information in order to determine whether his claim was valid or not. Help him.

Input Format

The input will start with an integer ($1 \leq T \leq 100$) representing the number of test cases. For each test case, a float point number will given representing the radius ($0 < R \leq 1{,}000{,}000$).

Output Format

For each test case, print the following line:

k. A

Where *k* is the test case number (starting at one) and *A* is the area of the largest rectangle that could be formed inside the given circle. The output should be rounded to the nearest two decimals places.

Sample Input / Output

farm.in

```
2
1
27.66
```

OUTPUT

```
1. 2.00
2. 1530.15
```

Johnny Love Matrices

Source file: magic.(cpp|java)
Input: magic.in
Balloon Color: Purple

Description

Even though Johnny hates math, when it comes to matrices, he just love dealing with them. He can spend hours and hours just checking the properties of these matrices. In fact, he has saved all the square matrices that he has ever used in a special file.

Lately, he read an article about Normal Magic Squares. Based on what he read, Normal Magic Square is an N x N matrix that contains all the integers from 1 to N^2 such that the sum of the elements in each of the N rows, and in each of the N columns, and in each of the two diagonals are the same. For example, the figure below shows the magic matrix with N = 3 where the sum of the elements across the rows, the columns, and the diagonals are 15.

8	1	6	15
3	5	7	15
4	9	2	15
15	15	15	

Johnny read also, that Normal Magic Squares exist for all orders N ≥ 1 except N = 2.

Being obsessed with matrices, Johnny decided to determine which of the matrices in his file is a Normal Magic Square? Since it is very hard to check those many matrices manually, he asked from you to write a program that would help him.

Given an N x N ($1 \leq N \leq 1{,}000$) matrix, your program should output whether the given matrix is a Normal Magic Square or not.

Input Format

The input will start with an integer *T* representing the number of test cases. For each test case, the first line consists of a single integer N representing the dimension of the matrix. Follows, N lines each having N integers representing the N x N matrix. All the elements in the matrix will be positive integers less than or equal to 1,000,000.

Output Format

For each test case, print the following line:

k. c

Where *k* is the test case number (starting at one) and *c* is either ‘T’ if the matrix is a Normal Magic Square or ‘F’ if the matrix is not a Normal Magic Square.

Sample Input / Output

magic.in

```
2
3
7 2 3
9 5 1
4 6 8
3
2 7 6
9 5 1
4 3 8
```

OUTPUT

```
1. F
2. T
```

Batman Begins

Source file: batman.(cpp|java)
Input: batman.in
Balloon Color: Green

Description

Ra's Al Ghul, head of the centuries-old League of Shadows, is an international terrorist and assassin whose ultimate goal is a world in perfect environmental balance. He believes that the best way to achieve this balance is to eliminate most of humanity. As the corruption increases in Gotham city, Ra's Al Ghul decided to destroy the city using a biological weapon. You, Batman, and your friend James Gordon decided to stop him. Ra's Al Ghul wants to use the train to get his genetically-engineered virus to the main water hub of Gotham city. In order to stop him, James Gordon wants to reach the main water hub before the train. Since you are the best programmer in Gotham city, Gordon asks from you to write a program that would calculate the minimum average speed V (in km/hr) that is required to reach the water hub before the train.

Input Format

The input will start with an integer ($1 \leq T \leq 100$) representing the number of test cases. For each test case, two integers will be given. The first integer ($1 \leq S \leq 3600$) represents the time (in seconds) that the train will take before reaching the water hub. The second integer ($1 \leq D \leq 50{,}000$) represents the distance (in meters) from the position of Gordon to the water hub.

Output Format

For each test case, print the following line:

k. V

Where k is the test case number (starting at one) and V is as explained in the description. The output should be rounded to the nearest two decimals places.

Sample Input / Output

batman.in

```
2
60 1000
55 500
```

OUTPUT

```
1. 60.00
2. 32.73
```

The Gardener

Source file: gardener{.c| .cpp| .java}

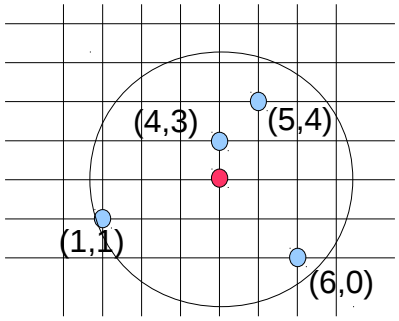
Input file: gardener.in

AUS gardeners are facing a problem! With the summer heat just around the corner, they have to hurry to install new water sprinklers for watering the plants in the campus gardens.

Each sprinkler has a radius R it can cover, which is selected during installation. If the radius selected is too big, water is wasted, so the idea is to find the location and minimum radius to use for each sprinkler.

Each garden, which is assumed to be a rectangular grid, will receive just one sprinkler. The dimensions of the grid are not supplied. Only the integer locations of the plants inside it. Each plant is also assumed to occupy a point on the 2D plane. This point must be within a sprinkler's R.

Each garden has at least three plants and the sprinkler has to be placed at integer coordinates for simplicity. Its radius though can be a floating point number. An example is given below:



Input File

The input file starts with a number M (M<100) describing how many cases are to be examined. Each case starts with a line containing the number N of plants (2<N<100), followed by N lines with the integer x and y coordinates of each plant (0=<x,y<=1000).

Output

For each input case, you should output in a separate line the location where the sprinkler should be placed (the x and y coordinates), and its radius with 2 decimal digits accuracy. Numbers should be separated by a single space.

Sample Input

```
2

4

1 1

4 3

5 4

6 0

3

1 1

2 2

3 3
```

Sample Output

```
4 2 3.16

                                    2 2 1.41
```