A - Calculator
Input File: CalculatorIn.txt

Baby Ryan is an arithmetic genius. Given any two integers, he can calculate their product, sum, difference and average. Your task is to write a program that is as smart as Ryan.

Inputs
There will be one line of input that contains two integers.

Outputs
There will be one line of output containing the two input numbers, followed by the integer product, sum, and difference of the two numbers, followed by the average of the two input integers formatted to one decimal place. The output line should be annotated exactly as shown below.

Sample inputs
10 15

Sample output
n1 = 10, n2 = 15, n1 * n2 = 150, n1 + n2 = 25, n1 - n2 = -5, the average of n1 and n2 = 12.5

B - Spring Break
Input File: SpringBreakIn.txt

Breanne is the treasurer of her class, and she and her classmates have just returned from their senior class trip. While away, the members of the class took turns paying the daily food bills, which were different from day to day. Now she would like make sure that the cost of all of the meals is shared equally among the students. Your task is to determine the *minimum* amount of money that has to be transferred between the students so that each of them spends the same amount of money on food, to the nearest penny.

Inputs

The first line of the input contains the number of class trips to consider. This will be followed by one data set for each trip, which consists of multiple lines. The first line of each data set contains one integer, n, which represents the number of students on the trip. The next n lines of each trip's data set contain the money each student spent on food during the trip, one student per line.

Outputs

There will be one line of output per class trip that contains the *minimum* amount of money that has to be transferred between all of the students so that each of them spends the same amount of money on food, to the nearest penny. The output should be formatted in standard US Currency format, as shown below.

Sample inputs	Sample output
2 3 10.0 20.0 30.0 5 1500.0 3.0 3.01 1500.01 255.34	$10.00 $1,695.47

C - Magic Numbers
Input File: PalindromeIn.txt

An integer palindrome is an integer that gives the same value when read from the left as when read from the right. Examples are 12321 and 1221. For *most* integers that are not palindromes, an integer palindrome can be generated from them by reversing the digits of the integer and adding the resulting integer to the original integer. If the resulting sum, `s`, is not a palindrome the process is repeated using the sum `s` as the integer non-palindrome until the sum is a palindrome. For example, for the integer non-palindrome 165 the process would proceed as follows:

```
  165        726        1353
  561        627        3531
 ----       ----       ----
  726       1353       4884
```

For any integer non-palindrome, `np`, if this process does not yield a palindrome that is less than 2,147,483,647 we will assume that the process cannot generate a palindrome for `np`.

Inputs
The first line of the input contains an integer, `n`, which is the number of integers to be converted to integer palindromes. That line will be followed by `n` lines, each of which contains a single integer to be converted to a palindrome.

Outputs
There will be `n` lines of output, one per number to be converted to a palindrome. If the number is a palindrome, it will be output. If it cannot be converted to a palindrome less than 2,147,483,647 the output will be: `no palindrome found` . (The period is not included as part of the output.) Otherwise, the line will contain the integer palindrome generated by the process.

Sample inputs	Sample output
6	4884
165	1221
1221	12321
12321	9339
195	no palindrome found
196	9366639
2656752	

D - Photos
Input File: PhotosIn.txt

Skyler would like to transfer the photos stored on the memory cards of her digital camera to her hard drive. Each photo consists of a rectangular grid of pixels (dots), and each pixel is stored as three binary numbers that specify the red, green, and blue color intensities (that when combined produce the color of the pixel). The minimum intensity of each color is zero, and the number of bits dedicated to each color is large enough to store the maximum intensity of the color. One problem: she has to catch the last train to Manhattan and may not have enough time to perform the transfer. Your task is to write a program that calculates the time required to transfer each memory card to her hard drive.

Inputs
The first line of input will contain the number of memory cards to transfer to Skyler's hard drive. This will be followed by a two line data set per memory card. The first line of each card's data set will contain four integers: the number of pictures on the card, followed by the card's data transfer rate in *bytes*-per-second, followed by the number of rows and the number of columns in the rectangular grid of pixels that make up every image on the card. The second line of each card's data set will contain three integers that represent the *maximum* intensity of each of the individual colors (red, green and blue) that make up each pixel on the card. All inputs on a line will be separated by a space.

Outputs
There will be one line of output per memory card that contains the total number of bits transferred followed by the number of seconds required to perform the transfer rounded to the nearest second. The two outputs will be separated by a space, and the number of bits transferred will always be less than 9,223, 372, 036, 854, 775, 807 $= (2^{63} - 1)$.

Sample inputs
```
3
10 20 30 40
15 15 15
200 100000 2000 1000
255 511 1023
200 100000 2000 1000
15 16 17
```

Sample outputs
```
144000 900
4341600000 54
5600000000 69
```

E - Justin Beaver
InputFile: JBIn.txt

The basis of Justin's Doctoral dissertation for his PhD in meteorology is his theory that beavers can solve the global warming problem. He has observed that in many years when the beaver population was high, the amount of CO_2 in the air was low. Having collected data sets from all over the world, he would like to automate his analysis. Each data set consists of many data points with each data point, e.g. (200 76), being a yearly beaver population followed by a measured CO_2 level. The Supportive Subsets of a data set are the subsets of the data set that contain the largest number of yearly data points in which the beaver population continually increases while the CO_2 level continuously decreases.

For example, the data set (15 50) (22 40) (75 88) (28 65) (12 30) (20 10) yields three Supportive Subsets each containing two data points: (15 50) (22 40), and (15 50) (20 10), and (12 30) (20 10). The first Supportive Subset is of "particular interest" to Justin because the beaver count of its first and last data point is equal to or higher than the first and last beaver count of any of the other Supportive Subsets. Your task is to determine the number of data points in the Supportive Subsets, the number of *different* data points that begin the Supportive Subsets and the data points that make up the subset of "particular interest".

Inputs
The first line of input will contain the number of data sets to consider. This will be followed by one group of inputs per data set, the first line of which will contain the number of data points in the data set. This will be followed by one line of input per data point containing the yearly beaver count followed by that year's CO_2 level. No two data points in a data set will contain the same number of beavers.

Outputs
For each data set there will be two lines of output. The first line will contain two integers which will be the number of data points in the Supportive Subsets, nMax, followed by the number of *different* data points that begin the Supportive Subsets. The second line will contain the nMax data points that make up the Supportive Subset of particular interest which begins *and* ends with the *highest* number of beavers arranged in ascending order based on the number of beavers. All outputs on a line should be separated by a space.

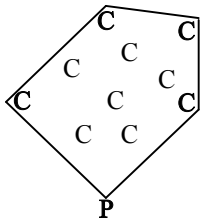
Sample inputs
```
3
6
15 50
22 40
75 88
28 65
12 30
20 10
3
200 20
100 10
300 30
10
750 2025
6008 1300
```

Sample inputs continued
```
5800 2175
500 2050
1000 4000
1100 4010
6000 2000
8000 1400
5600 2150
2000 1900
```

Sample outputs
```
2 2
15 50 22 40
1 3
300 30
4 3
1100 4010 5800 2175 6000 2000 8000 1400
```

F - Herding Chickens
Input File: HerdingChicksIn.txt

Chicken farmer Fern enjoys literature; however her author friend E. B. White once warned her that he didn't know which was "more discouraging, literature or chickens". Fern thinks chickens, and so she has decided to construct a fence around her chickens to keep them in place. The chickens have become very territorial, and each chick roosts at the same spot in the chicken yard every day. To determine the length of the fencing, she will tie a string to a post at or below the south edge chicken yard and then walk in a clockwise direction pulling the string taught to encircle all of the chickens until she returns to the post (see below). Your task will be to determine the length of the string given the location of the post and each chicken's location.



C

_____ = string

C = chicken

P = post

Inputs

The first line of the input contains the number of chicken yards to enclose. This is followed by one data set for each yard. The first line of a data set contains two real numbers that are the x and y coordinates (in feet) of the post. The next line contains one integer, n, that is the number of chickens roosting in the yard. This is followed by one line per chicken. Each of these n lines contains two real numbers, which are the x and y coordinates (in feet) of a roosting chicken. Since the post is at or below the south edge of the yard, the y coordinate of the post is always less than the y coordinates of the chickens.

Outputs

There will be one line of output per chicken yard. This line will contain length of the string, in feet, that encircles all the chickens in the yard rounded to two decimal places.

Sample inputs	Sample output
2	160.00
0.0 0.0	113.14
6	
-30.0 20.0	
-30.0 0.0	
-20.0 5.0	
30.0 0.0	
1.0 1.0	
30.0 20.0	
0.0 -20.0	
6	
0.0 20.0	
10.0 5.0	
20.0 0.0	
5.0 -10.0	
-20.0 0.0	
5.0 10.0	

G - Binary University
Input File: BinaryUIn.txt

Nora attends Binary University and is about to register for her spring courses. Every course – except for terminal courses – is a prerequisite for *one or two* other courses, which is how BU got its name. In order to graduate, Nora must complete several study tracks. Each study track begins with an introductory course that all students beginning the track must take. After taking a course, they must select one, and only one, of its follow-up courses (courses for which it is a prerequisite). Students cannot take a course in a study track unless they have taken its prerequisite. This course selection process continues until they complete the track by taking a terminal course (i.e., one that is not a prerequisite for any other course). Every course has a course number, with the more difficult courses having *higher* numbers.

To increase her chances for admission to medical school, Nora has decided to take the most difficult study path through the pre-med track, which is the sequence of courses that when their course numbers are added together produces the *highest* total. Your task is to determine that total, and the course numbers of the courses that make up that total.

Inputs

The first line of input will contain the number of pre-med tracks to consider, which will be followed by one data set per track. The first line of each data set will contain the number of courses in the track. This will be followed by one course description line per course, the first of which describes the introductory course. Each course description line will containing three integers separated by a space. The first integer is the course number, and the next two integers are the course numbers of its two follow-up courses. A follow-up course number of zero indicates the follow-up course does not exist.

Outputs

There will be two lines of output for each pre-med track considered. The first line will contain the total of the course numbers that make up the most difficult study path through the track. The second line of output will give the courses numbers that make up the most difficult study path, in prerequisite order.

Sample Inputs

```
2
5
100 200 700
700 0 0
200 221 210
221 0 0
210 0 0
6
100 104 150
104 187 0
150 160 159
160 0 0
187 0 0
159 0 0
```

Sample Output

```
800
100 700
410
100 150 160
```