Problem A
Height Ordering
Problem ID: height

Mrs. Chambers always has her class line up in height order (shortest at the front of the line). Every September a new class of exactly 20 3rd graders arrive, all of different height. For the first few days it takes a long time to get the kids in height order, since no one knows where they should be in the line. Needless to say, there is quite a bit of jockeying around. This year Mrs. Chambers decided to try a new method to minimize this ordering chaos. One student would be selected to be the first person in line. Then, another student is selected and would find the *first* person in the line that is taller than him, and stand in front of that person, thereby causing all the students behind him to step back to make room. If there is no student that is taller, then he would stand at the end of the line. This process continues, one student at-a-time, until all the students are in line, at which point the students will be lined up in height order.

For this problem, you will write a program that calculates the total number of steps taken back during the ordering process for a given class of students.

Input

The first line of input contains a single integer P, ($1 \leq P \leq 1000$) which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, K, followed by 20 non-negative unique integers separated by a single space. The 20 integers represent the height (in millimeters) of each student in the class.

Output

For each data set there is one line of output. The single output line consists of the data set number, K, followed by a single space followed by total number of steps taken back.

Sample Input 1

```
4
1 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919
2 919 918 917 916 915 914 913 912 911 910 909 908 907 906 905 904 903 902 901 900
3 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 900
4 918 917 916 915 914 913 912 911 910 909 908 907 906 905 904 903 902 901 900 919
```

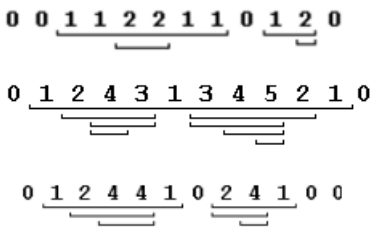
Sample Output 1

```
1 0
2 190
3 19
4 171
```

This page is intentionally left blank.

Problem B
Islands in the Data Stream
Problem ID: islands

Given a sequence of integers $a_1, a_2, a_3, \ldots, a_n$, an island in the sequence is a contiguous subsequence for which each element is greater than the elements immediately before and after the subsequence. In the examples below, each island in the sequence has a bracket below it. The bracket for an island contained within another island is below the bracket of the containing island.



Write a program that takes as input a sequence of 12 non-negative integers and outputs the number of islands in the sequence.

Input

The first line of input contains a single integer P, ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, K, followed by 12 non-negative integers separated by a single space. The first and last integers in the sequence will be 0.

Output

For each data set there is one line of output. The single output line consists of the data set number, K, followed by a single space followed by the number of islands in the sequence.

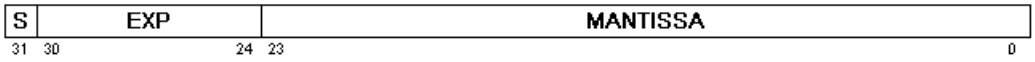
Sample Input 1	Sample Output 1
4 1 0 0 1 1 2 2 1 1 0 1 2 0 2 0 1 2 4 3 1 3 4 5 2 1 0 3 0 1 2 4 4 1 0 2 4 1 0 0 4 0 1 2 3 4 5 6 7 8 9 10 0	1 4 2 8 3 6 4 10

This page is intentionally left blank.

Problem C
Floating-Point Format Conversion
Problem ID: conversion

To help support a patent defense, we need to recover some experimental data that was stored as single precision floating point on a now-defunct Gould Power-Node mini-computer. The Gould used a base 16 floating-point format. We want to convert Gould floating point values, as much as possible, to single precision *IEEE* floating-point values.

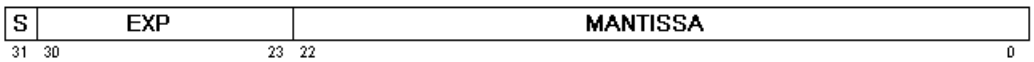
The Gould internal floating-point format has 1 sign bit, S, a 7-bit offset (base 16) exponent field, E, and a 24-bit (6 hex digits) hexadecimal mantissa. (Note that this means that up to 3 high bits of the mantissa may be zero.)



$$\text{value} = ((-1)^{S})((16)^{(EXP-64)})(MANTISSA / (2^{24}))$$

Floating-point zero is represented by 32 bits of 0.

The IEEE format has 1 sign bit, S, an 8-bit offset (base 2) exponent field, E, and a 24-bit mantissa, for which the high bit is (in normalized numbers) always 1 and not part of the 23 bits in the format.



If the exponent is not 255 and not 0, the value is a normalized floating point number,

$$\text{value} = ((-1)^{S})((2)^{(EXP-127)})(1 + (MANTISSA / (2^{23})))$$

If the exponent is 255 and the mantissa is 0, the value is plus or minus infinity (depending on the sign bit). If the exponent is 255 and the mantissa is not 0, it indicates special values that will not be used in this problem.

If the exponent is 0 and the mantissa is zero, the value is plus or minus zero (depending on the sign bit).

If the exponent is 0 and the mantissa is not zero, the value is a de-normalized floating-point number with:

$$\text{value} = ((-1)^{S})((2)^{(-126)})(MANTISSA / (2^{23}))$$

Write a program that takes as input a floating-point value in Gould format and outputs the value in IEEE format as follows:

If the value is zero return (plus) zero.

If the value is too large to be represented as a normalized floating-point value, return plus or minus infinity depending on the sign.

If the value is too small to be represented as a normalized floating-point value:

- If it may be represented as a de-normalized value, return the de-normalized value.
- Otherwise, return plus or minus zero, depending on the sign.

In all other cases, return the normalized value.

If there are less significant bits than required for IEEE floating-point, extend with 0 bits.

If there are more significant bits than required for IEEE floating-point, truncate the extra bits.

Input

The first line of input contains a single integer P, ($1 \leq P \leq 1000$) which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, K, followed by 8 hex digits (`0-9,A-F`) of the Gould floating-point value.

Output

For each data set there is one line of output. The single output line consists of the data set number, K, followed by a single space followed by the 8 hex digits (`0-9,A-F`) of the corresponding (as described above) IEEE floating point value.

Sample Input 1	Sample Output 1
4 1 41200000 2 E0FFFFFE 3 E11FFFFF 4 88888888	1 40000000 2 FF7FFFFE 3 FF800000 4 80000000

Problem D
Happy Happy Prime Prime
Problem ID: happyprime

RILEY VASHTEE: [*reading from display*] Find the next number in the sequence:

313 331 367 ...? What?

THE DOCTOR: 379.

MARTHA JONES: What?

THE DOCTOR: It's a sequence of happy primes – **379**.

MARTHA JONES: Happy *what?*

THE DOCTOR: Any number that reduces to one when you take the sum of the square of its digits and continue iterating it until it yields 1 is a happy number. Any number that doesn't, isn't. A *happy* prime is both happy and prime.

THE DOCTOR: I dunno, talk about *dumbing down*. Don't they teach recreational mathematics anymore?

Excerpted from "*Dr. Who*," Episode 42 (2007).

The number 7 is certainly prime. But is it happy?

$$\begin{aligned} 7 &\rightarrow 7^2 = 49 \\ 49 &\rightarrow 4^2 + 9^2 = 97 \\ 97 &\rightarrow 9^2 + 7^2 = 130 \\ 130 &\rightarrow 1^2 + 3^2 = 10 \\ 10 &\rightarrow 1^2 + 0^2 = 1 \end{aligned}$$

It is happy :-) As it happens, 7 is the smallest happy prime. Please note that for the purposes of this problem, 1 is *not* prime.

For this problem you will write a program to determine if a number is a happy prime.

Input

The first line of input contains a single integer P, ($1 \leq P \leq 10000$), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, K, followed by the happy prime candidate, m, ($1 \leq m \leq 10000$).

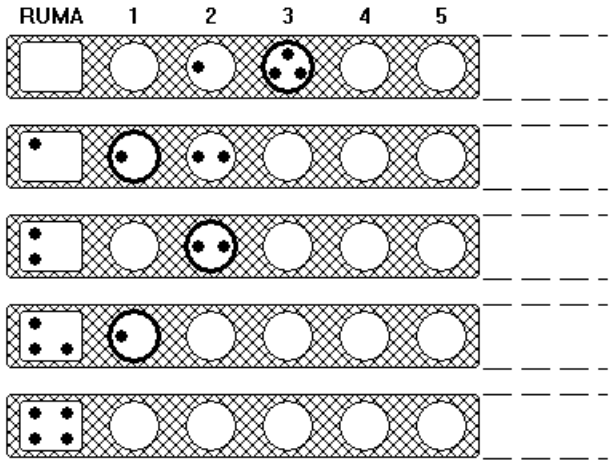
Output

For each data set there is a single line of output. The single output line consists of the data set number, K, followed by a single space followed by the candidate, m, followed by a single space, followed by `YES` or `NO`, indicating whether m is a happy prime.

Sample Input 1	Sample Output 1
4 1 1 2 7 3 383 4 1000	1 1 NO 2 7 YES 3 383 YES 4 1000 NO

Problem E
Mancala
Problem ID: mancala

Mancala is a family of board games played around the world, sometimes called *sowing* games, or *count-and-capture* games, which describes the game play. One simple variant is a solitaire game called *Tchoukaillon* which was described by Véronique Gautheron. *Tchoukaillon* is played on a board with an arbitrary number of bins numbered $1, 2, \ldots$, containing $b[1], b[2], ...$, counters respectively and an extra empty bin called the Roumba on the left.



A single play consists on choosing a bin, n, for which $b[n] = n$ (indicated by the darker circles in the diagram) and distributing the counters one per bin to the bins to the left including the *Roumba* (getting the next diagram below in the figure above). If there is no bin where $b[n] = n$, then the board is a losing board.

If there is a sequence of plays which takes the initial board distribution to one in which every counter is in the *Roumba*, the initial distribution is called a winnable board. In the example above, $0, 1, 3, \ldots$ is a winnable board (the "..." indicates all the bins to the right of bin 3 contain 0). For each total number of counters, there is a unique distribution of the counters to bins to make a winnable board for that total count (so $0, 1, 3, \ldots$ is the only winnable board with 4 counters).

Write a program which finds the winnable board for a total count input.

Input

The first line of input contains a single integer P, ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, K, followed by a single space, followed by the total count N ($1 \leq N \leq 2000$) of the winnable board to be found.

Output

For each data set there will be multiple lines of output. The first line of output contains the data set number, K, followed by a single space, followed by the index of the last bin, B, with a non-zero count. Input will be chosen so that B will be no more than 80. The first line of output for each dataset is followed by the bin counts $b[1], b[2], \ldots, b[B]$, 10 per line separated by single spaces.

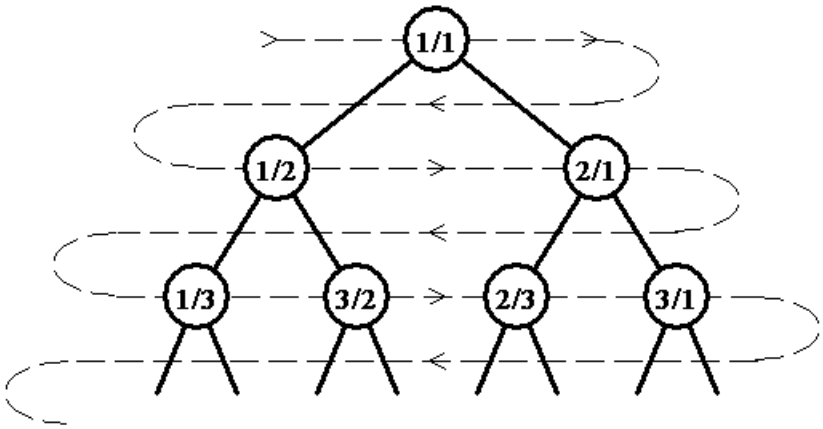
Sample Input 1	Sample Output 1
3 1 4 2 57 3 500	1 3 0 1 3 2 12 1 2 2 2 2 6 2 4 6 8 10 12 3 39 0 2 2 1 3 2 2 2 6 7 5 0 6 12 2 6 10 14 18 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39

Problem F
A Rational Sequence
Problem ID: sequence

An infinite full binary tree labeled by positive rational numbers is defined by:

- The label of the root is `1/1`.
- The left child of label `p/q` is `p/(p+q)`.
- The right child of label `p/q` is `(p+q)/q`.

The top of the tree is shown in the following figure:



A rational sequence is defined by doing a level order (breadth first) traversal of the tree (indicated by the light dashed line). So that:

```
F(1) = 1/1, F(2) = 1/2, F(3) = 2/1, F(4) = 1/3, F(5) = 3/2, F(6) = 2/3, ...
```

Write a program which takes as input a rational number, `p/q`, in lowest terms and finds the next rational number in the sequence. That is, if `F(n) = p/q`, then the result is `F(n+1)`.

Input

The first line of input contains a single integer P, ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, K, which is then followed by a space, then the numerator of the fraction, p, followed immediately by a forward slash (/), followed immediately by the denominator of the fraction, q. Both p and q will be relatively prime and $0 \leq p, q \leq 2,147,483,647$.

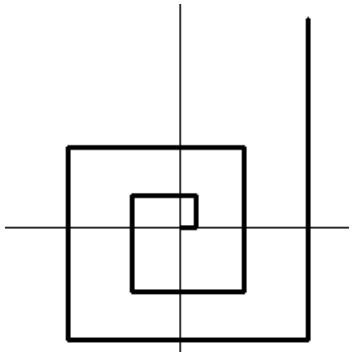
Output

For each data set there is a single line of output. It contains the data set number, K, followed by a single space which is then followed by the numerator of the fraction, followed immediately by a forward slash (/) followed immediately by the denominator of the fraction. Inputs will be chosen such that neither the numerator nor the denominator will overflow a 32-bit integer.

Sample Input 1	Sample Output 1
5 1 1/1 2 1/3 3 5/2 4 2178309/1346269 5 1/10000000	1 1/2 2 3/2 3 2/5 4 1346269/1860498 5 10000000/9999999

Problem G
Growing Rectangular Spiral
Problem ID: spiral

A growing rectangular spiral is a connected sequence of straight-line segments starting at the origin. The first segment goes right (positive x direction). The next segment goes up (positive y direction). The next segment goes left (negative x direction). The next segment goes down (negative y direction) and the sequence of directions repeats. Each segment has integer length and each segment is at least one unit longer than the previous segment. In the spiral below, the segment lengths are 1, 2, 4, 6, 7, 9, 11, 12, 15, 20.



Write a program to determine the shortest growing rectangular spiral (in total length) that ends at a given integer point (x, y) in the first quadrant or determine that there is no such spiral.

Input

The first line of input contains a single integer P, ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input consisting of three space separated decimal integers. The first integer is the data set number. The next two integers are the x and y coordinates of the desired end point ($1 \leq x \leq 10000, 1 \leq y \leq 10000$).

Output

For each data set there is a single line of output. If there is no spiral solution, the line consists of the data set number, a single space and `"NO PATH"` (without the quotes). If there is a solution, the line consists of the data set number, a single space, the number of segments in the solution, a single space, followed by the lengths of the segments in order, separated by single spaces. The input data will be chosen so that no path requires more than 22 segments.

Sample Input 1	Sample Output 1
3 1 1 1 2 3 5 3 8 4	1 NO PATH 2 2 3 5 3 6 1 2 3 9 10 11

This page is intentionally left blank.

Problem H
Farey Sums
Problem ID: fareysums

Given a positive integer, N, the sequence of all fractions a/b with $(0 < a \leq b), (1 \leq b \leq N)$ and a and b relatively prime, listed in increasing order, is called the *Farey Sequence of order N*.

For example, the *Farey Sequence of order 6* is:

$$\frac{0}{1}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{5}{6}, \frac{1}{1}$$

If the denominators of the *Farey Sequence of order N* are:

$$b_1, b_2, \ldots, b_K$$

then the *Farey Sum of order N* is the sum of b_i/b_{i+1} from $i = 1 \ldots K - 1$.

For example, the *Farey Sum of order 6* is:

$$\frac{1}{6} + \frac{6}{5} + \frac{5}{4} + \frac{4}{3} + \frac{3}{5} + \frac{5}{2} + \frac{2}{5} + \frac{5}{3} + \frac{3}{4} + \frac{4}{5} + \frac{5}{6} + \frac{6}{1} = \frac{35}{2}$$

Write a program to compute the *Farey Sum of order N* (input)!

Input

The first line of input contains a single integer P, $(1 \leq P \leq 9999)$, which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, K, followed by the order N, $(2 \leq N \leq 10000)$, of the *Farey Sum* that is to be computed.

Output

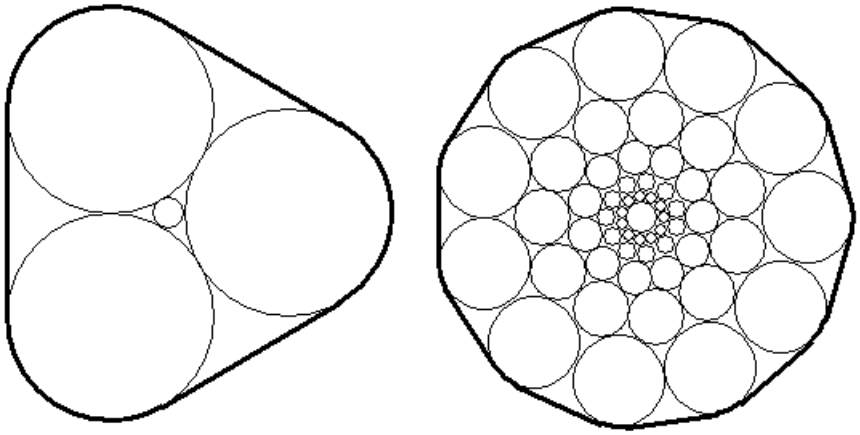
For each data set there is a single line of output. The single output line consists of the data set number, K, followed by a single space followed by the *Farey Sum* as a decimal fraction in lowest terms. If the denominator is 1, print only the numerator.

Sample Input 1	Sample Output 1
4 1 6 2 15 3 57 4 9999	1 35/2 2 215/2 3 2999/2 4 91180457/2

This page is intentionally left blank.

Problem I
The Queen's Super-circular Patio
Problem ID: queenspatio

The queen wishes to build a patio paved with of a circular center stone surrounded by circular rings of circular stones. All the stones in a ring will be the same size with the same number of stones in each ring. The stones in the innermost ring will be placed touching (tangent to) the adjacent stones in the ring and the central stone. The stones in the other rings will touch the two adjacent stones in the next inner ring and their neighbors in the same ring. The figures below depict a patio with one ring of three stones and a patio with 5 rings of 11 stones. The patio is to be surrounded by a fence that goes around the outermost stones and straight between them (the heavier line in the figures).



The queen does not yet know how many stones there will be in each circle nor how many circles of stones there will be. To be prepared for whatever she decides, write a program to calculate the sizes of the stones in each circle and the length of the surrounding fence. The radius of the central stone is to be one *queenly* foot.

Input

The first line of input contains a single integer P, ($1 \leq P \leq 1000$) which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input. It contains the data set number, K, the number, N ($3 \leq N \leq 20$), of stones in each circle and the number, M ($1 \leq M \leq 15$), of circles of stones around the central stone.

Output

For each data set there is a single line of output. It contains the data set number, K, followed by a single space which is then followed by the radius (in queenly feet) of the stones in the outermost ring (to 3 decimal places) which is followed by a single space which is then followed by the length (in *queenly feet*) of the fence (to 3 decimal places).

Sample Input 1	Sample Output 1
3 1 3 1 2 7 3 3 11 5	1 6.464 79.400 2 3.834 77.760 3 2.916 82.481