1 Problem A: Fermat vs. Pythagoras

Background

Computer generated and assisted proofs and verification occupy a small niche in the realm of Computer Science. The first proof of the four-color problem was completed with the assistance of a computer program and current efforts in verification have succeeded in verifying the translation of high-level code down to the chip level.

This problem deals with computing quantities relating to part of Fermat's Last Theorem: that there are no integer solutions of $a^n + b^n = c^n$ for $n > 2$.

The Problem

Given a positive integer N, you are to write a program that computes two quantities regarding the solution of

$$x^2 + y^2 = z^2$$

where x, y, and z are constrained to be positive integers less than or equal to N. You are to compute the number of triples (x, y, z) such that x, y, and z are relatively prime, i.e., have no common divisor larger than 1. You are also to compute the number of values $0 < p \leq N$ such that p is not part of any triple (not just relatively prime triples).

The Input

The input consists of a sequence of positive integers, one per line. Each integer in the input file will be less than or equal to 1,000. Input is terminated by end-of-file.

The Output

For each integer N in the input file print two integers separated by a space. The first integer is the number of relatively prime triples (such that each component of the triple is $\leq N$). The second number is the number of positive integers $\leq N$ that are not part of any triple whose components are all $\leq N$. There should be one output line for each input line.

Sample Input

```
10
25
100
```

Sample Output

```
1 4
4 9
16 27
```

2 Problem B: The Cat in the Hat

Background

(An homage to Theodore Seuss Geisel)

> The Cat in the Hat is a nasty creature,
> But the striped hat he is wearing has a rather nifty feature.
>
> With one flick of his wrist he pops his top off.
>
> Do you know what's inside that Cat's hat?
> A bunch of small cats, each with its own striped hat.
>
> Each little cat does the same as line three,
> All except the littlest ones, who just say "Why me?"
>
> Because the littlest cats have to clean all the grime,
> And they're tired of doing it time after time!

The Problem

A clever cat walks into a messy room which he needs to clean. Instead of doing the work alone, it decides to have its helper cats do the work. It keeps its (smaller) helper cats inside its hat. Each helper cat also has helper cats in its own hat, and so on. Eventually, the cats reach a smallest size. These smallest cats have no additional cats in their hats. These unfortunate smallest cats have to do the cleaning.

The number of cats inside each (non-smallest) cat's hat is a constant, N. The height of these cats-in-a-hat is $\frac{1}{N+1}$ times the height of the cat whose hat they are in.

> The smallest cats are of height one;
> these are the cats that get the work done.

All heights are positive integers.

Given the height of the initial cat and the number of worker cats (of height one), find the number of cats that are not doing any work (cats of height greater than one) and also determine the sum of all the cats' heights (the height of a stack of all cats standing one on top of another).

The Input

The input consists of a sequence of cat-in-hat specifications. Each specification is a single line consisting of two positive integers, separated by white space. The first integer is the height of the initial cat, and the second integer is the number of worker cats.

A pair of 0's on a line indicates the end of input.

The Output

For each input line (cat-in-hat specification), print the number of cats that are not working, followed by a space, followed by the height of the stack of cats. There should be one output line for each input line other than the "0 0" that terminates input.

Sample Input

```
216 125
5764801 1679616
0 0
```

Sample Output

```
31 671
335923 30275911
```

3 Problem C: Maximum Sum

Background

A problem that is simple to solve in one dimension is often much more difficult to solve in more than one dimension. Consider satisfying a boolean expression in conjunctive normal form in which each conjunct consists of exactly 3 disjuncts. This problem (3-SAT) is NP-complete. The problem 2-SAT is solved quite efficiently, however.

In contrast, some problems belong to the same complexity class regardless of the dimensionality of the problem.

The Problem

Given a 2-dimensional array of positive and negative integers, find the sub-rectangle with the largest sum. The sum of a rectangle is the sum of all the elements in that rectangle. In this problem the sub-rectangle with the largest sum is referred to as the *maximal sub-rectangle*.

A sub-rectangle is any contiguous sub-array of size 1×1 or greater located within the whole array. As an example, the maximal sub-rectangle of the array:

$$\begin{array}{rrrr} 0 & -2 & -7 & 0 \\ 9 & 2 & -6 & 2 \\ -4 & 1 & -4 & 1 \\ -1 & 8 & 0 & -2 \end{array}$$

is in the lower-left-hand corner:

$$\begin{array}{rr} 9 & 2 \\ -4 & 1 \\ -1 & 8 \end{array}$$

and has the sum of 15.

The Input

The input consists of an $N \times N$ array of integers.

The input begins with a single positive integer N on a line by itself indicating the size of the square two dimensional array. This is followed by N^2 integers separated by white-space (newlines and spaces). These N^2 integers make up the array in row-major order (i.e., all numbers on the first row, left-to-right, then all numbers on the second row, left-to-right, etc.). N may be as large as 100. The numbers in the array will be in the range $[-127, 127]$.

The Output

The output is the sum of the maximal sub-rectangle.

Sample Input

```
4
0 -2 -7  0 9  2 -6  2
-4  1 -4  1 -1
8  0 -2
```

Sample Output

```
15
```

4 Problem D: SCUD Busters

Background

Some problems are difficult to solve but have a simplification that is easy to solve. Rather than deal with the difficulties of constructing a model of the Earth (a somewhat oblate spheroid), consider a pre-Columbian flat world that is a 500 kilometer $\times$ 500 kilometer square.

In the model used in this problem, the flat world consists of several warring kingdoms. Though warlike, the people of the world are strict isolationists; each kingdom is surrounded by a high (but thin) wall designed to both protect the kingdom and to isolate it. To avoid fights for power, each kingdom has its own electric power plant.

When the urge to fight becomes too great, the people of a kingdom often launch missiles at other kingdoms. Each SCUD missile (Sanitary Cleansing Universal Destroyer) that lands within the walls of a kingdom destroys that kingdom's power plant (without loss of life).

The Problem

Given coordinate locations of several kingdoms (by specifying the locations of houses and the location of the power plant in a kingdom) and missile landings you are to write a program that determines the total area of all kingdoms that are without power after an exchange of missile fire.

In the simple world of this problem kingdoms do not overlap. Furthermore, the walls surrounding each kingdom are considered to be of zero thickness. The wall surrounding a kingdom is the minimal-perimeter wall that completely surrounds all the houses and the power station that comprise a kingdom; the area of a kingdom is the area enclosed by the minimal-perimeter thin wall.

There is exactly one power station per kingdom.

There may be empty space between kingdoms.

The Input

The input is a sequence of kingdom specifications followed by a sequence of missile landing locations.

A kingdom is specified by a number N ($3 \leq N \leq 100$) on a single line which indicates the number of sites in this kingdom. The next line contains the x and y coordinates of the power station, followed by $N-1$ lines of x, y pairs indicating the locations of homes served by this power station. A value of -1 for N indicates that there are no more kingdoms. There will be at least one kingdom in the data set.

Following the last kingdom specification will be the coordinates of one or more missile attacks, indicating the location of a missile landing. Each missile location is on a line by itself. You are to process missile attacks until you reach the end of the file.

Locations are specified in kilometers using coordinates on a 500 km by 500 km grid. All coordinates will be integers between 0 and 500 inclusive. Coordinates are specified as a pair of integers separated by white-space on a single line. The input file will consist of up to 20 kingdoms, followed by any number of missile attacks.

The Output

The output consists of a single number representing the total area of all kingdoms without electricity after all missile attacks have been processed. The number should be printed with (and correct to) two decimal places.

Sample Input

```
12
3 3
4 6
4 11
4 8
10 6
5 7
6 6
6 3
7 9
10 4
10 9
1 7
5
20 20
20 40
40 20
40 40
30 30
3
10 10
21 10
21 13
-1
5 5
20 12
```

Sample Output

```
70.50
```

A Hint

You may or may not find the following formula useful.

Given a polygon described by the vertices $v_0, v_1, \ldots, v_n$ such that $v_0 = v_n$, the signed area of the polygon is given by

$$a = \frac{1}{2} \sum_{i=1}^{n} (x_{i-1} y_i) - (x_i y_{i-1})$$

where the x, y coordinates of $v_i = (x_i, y_i)$; the edges of the polygon are from v_i to v_{i+1} for $i = 0 \ldots n-1$.

If the points describing the polygon are given in a counterclockwise direction, the value of a will be positive, and if the points of the polygon are listed in a clockwise direction, the value of a will be negative.

5 Problem E: Meta-Loopless Sorts

Background

Sorting holds an important place in computer science. Analyzing and implementing various sorting algorithms forms an important part of the education of most computer scientists, and sorting accounts for a significant percentage of the world's computational resources. Sorting algorithms range from the bewilderingly popular Bubble sort, to Quicksort, to parallel sorting algorithms and sorting networks. In this problem you will be writing a program that creates a sorting program (a meta-sorter).

The Problem

The problem is to create a program whose output is a standard Pascal program that sorts n numbers where n is the only input to the program you will write. The Pascal program generated by your program must have the following properties:

- It must begin with `program sort(input,output);`
- It must declare storage for exactly n `integer` variables. The names of the variables must come from the first n letters of the alphabet (a,b,c,d,e,f).
- A single `readln` statement must read in values for all the integer variables.
- Other than `writeln` statements, the only statements in the program are `if then else` statements. The boolean conditional for each `if` statement must consist of one strict inequality (either $<$ or $>$) of two integer variables. Exactly $n!$ `writeln` statements must appear in the program.
- Exactly three semi-colons must appear in the program
  1. after the program header: `program sort(input,output);`
  2. after the variable declaration: `...:  integer;`
  3. after the `readln` statement: `readln(...);`
- No redundant comparisons of integer variables should be made. For example, during program execution, once it is determined that $a < b$, variables a and b should not be compared again.
- Every `writeln` statement must appear on a line by itself.
- The program must compile. Executing the program with input consisting of any arrangement of any n distinct integer values should result in the input values being printed in sorted order.

For those unfamiliar with Pascal syntax, the example at the end of this problem completely defines the small subset of Pascal needed.

The Input

The input is a single integer n on a line by itself with $1 \leq n \leq 6$.

The Output

The output is a compilable standard Pascal program meeting the criteria specified above.

Sample Input

3

Sample Output

```
program sort(input,output);
var
a,b,c : integer;
begin
  readln(a,b,c);
  if a < b then
    if b < c then
      writeln(a,b,c)
    else if a < c then
      writeln(a,c,b)
    else
      writeln(c,a,b)
  else
    if a < c then
      writeln(b,a,c)
    else if b < c then
      writeln(b,c,a)
    else
      writeln(c,b,a)
end.
```

6 Problem F: History Grading

Background

Many problems in Computer Science involve maximizing some measure according to constraints.

Consider a history exam in which students are asked to put several historical events into chronological order. Students who order all the events correctly will receive full credit, but how should partial credit be awarded to students who incorrectly rank one or more of the historical events?

Some possibilities for partial credit include:

1. 1 point for each event whose rank matches its correct rank
2. 1 point for each event in the longest (not necessarily contiguous) sequence of events which are in the correct order relative to each other.

For example, if four events are correctly ordered 1 2 3 4 then the order 1 3 2 4 would receive a score of 2 using the first method (events 1 and 4 are correctly ranked) and a score of 3 using the second method (event sequences 1 2 4 and 1 3 4 are both in the correct order relative to each other).

In this problem you are asked to write a program to score such questions using the second method.

The Problem

Given the correct chronological order of n events $1, 2, \ldots, n$ as $c_1, c_2, \ldots c_n$ where $1 \leq c_i \leq n$ denotes the ranking of event i in the correct chronological order and a sequence of student responses $r_1, r_2, \ldots r_n$ where $1 \leq r_i \leq n$ denotes the chronological rank given by the student to event i; determine the length of the longest (not necessarily contiguous) sequence of events in the student responses that are in the correct chronological order relative to each other.

The Input

The first line of the input will consist of one integer n indicating the number of events with $2 \leq n \leq 20$. The second line will contain n integers, indicating the correct chronological order of n events. The remaining lines will each consist of n integers with each line representing a student's chronological ordering of the n events. All lines will contain n numbers in the range $[1 \ldots n]$, with each number appearing exactly once per line, and with each number separated from other numbers on the same line by one or more spaces.

The Output

For each student ranking of events your program should print the score for that ranking. There should be one line of output for each student ranking.

Sample Input 1

```
4
4 2 3 1
1 3 2 4
3 2 1 4
2 3 4 1
```

Sample Output 1

```
1
2
3
```

Sample Input 2

```
10
3 1 2 4 9 5 10 6 8 7
1 2 3 4 5 6 7 8 9 10
4 7 2 3 10 6 9 1 5 8
3 1 2 4 9 5 10 6 8 7
2 10 1 3 8 4 9 5 7 6
```

Sample Output 2

```
6
4
10
5
```