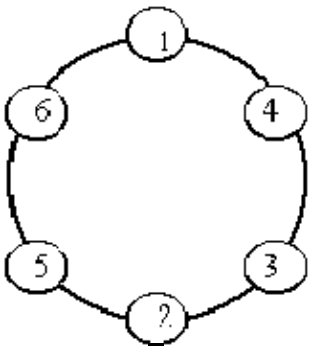
524 - Prime Ring Problem

Time limit: 3.000 seconds

A ring is composed of n (even number) circles as shown in diagram. Put natural numbers $1, 2, \ldots, n$ into each circle separately, and the sum of numbers in two adjacent circles should be a prime.



Note: the number of first circle should always be 1.

Input

n (0 < n <= 16)

Output

The output format is shown as sample below. Each row represents a series of circle numbers in the ring beginning from 1 clockwisely and anticlockwisely. The order of numbers must satisfy the above requirements.

You are to write a program that completes above process.

Sample Input

```
6
8
```

Sample Output

```
Case 1:
1 4 3 2 5 6
1 6 5 2 3 4

Case 2:
1 2 3 8 5 6 7 4
1 2 5 8 3 4 7 6
1 4 7 6 5 8 3 2
1 6 7 4 3 8 5 2
```

10344 - 23 out of 5

Time limit: 3.000 seconds

Your task is to write a program that can decide whether you can find an arithmetic expression consisting of five given numbers a_i (1<=i<=5) that will yield the value 23.
For this problem we will only consider arithmetic expressions of the following from:

$$(((a_{\pi(1)}\ o_1\ a_{\pi(2)})\ o_2\ a_{\pi(3)})\ o_3\ a_{\pi(4)})\ o_4\ a_{\pi(5)}$$

where π: {1,2,3,4,5} -> {1,2,3,4,5} is a bijective function
and $o_i \in$ {+,-,*} (1<=i<=4)

Input

The Input consists of 5-Tupels of positive Integers, each between 1 and 50.
Input is terminated by a line containing five zero's. This line should not be processed.

Output

For each 5-Tupel print "Possible" (without quotes) if their exists an arithmetic expression (as described above) that yields 23. Otherwise print "Impossible".

Sample Input

```
1 1 1 1 1
1 2 3 4 5
2 3 5 7 11
0 0 0 0 0
```

Sample Output

```
Impossible
Possible
Possible
```

10245 - The Closest Pair Problem

Time limit: 3.000 seconds

Given a set of points in a two dimensional space, you will have to find the distance between the closest two points.

Input

The input file contains several sets of input. Each set of input starts with an integer **N (0<=N<=10000)**, which denotes the number of points in this set. The next **N** line contains the coordinates of **N** two-dimensional points. The first of the two numbers denotes the **X-coordinate** and the latter denotes the **Y-coordinate**. The input is terminated by a set whose **N=0**. This set should not be processed. The value of the coordinates will be less than **40000** and non-negative.

Output

For each set of input produce a single line of output containing a floating point number (with four digits after the decimal point) which denotes the distance between the closest two points. If there is no such two points in the input whose distance is less than **10000**, print the line **INFINITY**.

Sample Input

```
3
0 0
10000 10000
20000 20000
5
0 2
6 67
43 71
39 107
189 140
0
```

Sample Output

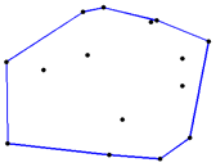
```
INFINITY
36.2215
```

(World Final Warm-up Contest, Problem setter: Shahriar Manzoor)

"Generally, a brute force method has only two kinds of reply, a) Accepted b) Time Limit Exceeded."

11626 - Convex Hull

Time limit: 1.000 seconds



Finding the convex hull of a set of points is an important problem that is often part of a larger problem. There are many algorithms for finding the convex hull. Since problems involving the convex hull sometimes appear in the ACM World Finals, it is a good idea for contestants to know some of these algorithms.

Finding the convex hull of a set of points in the plane can be divided into two sub-tasks. First, given a set of points, find a subset of those points that, when joined with line segments, form a convex polygon that encloses all of the original points. Second, output the points of the convex hull in order, walking counter-clockwise around the polygon. In this problem, the first sub-task has already been done for you, and your program should complete the second sub-task. That is, given the points that are known to lie on the convex hull, output them in order walking counter-clockwise around the hull.

Input Specification

The first line of input contains a single integer, the number of test cases to follow. The first line of each test case contains a single integer 3 <= *n* <= 100000, the number of points. The following *n* lines of the test case each describe a point. Each of these lines contains two integers and either a `Y` or an `N`, separated by spaces. The two integers specify the x- and y-coordinates of the point. A `Y` indicates that the point is on the convex hull of all the points, and a `N` indicates that it is not. The x- and y-coordinates of each point will be no less than -1000000000 and no greater than 1000000000. No point will appear more than once in the same test case. The points in a test case will never all lie on a line.

Sample Input

```
1
5
1 1 Y
1 -1 Y
0 0 N
-1 -1 Y
-1 1 Y
```

Output Specification

For each test case, generate the following output. First, output a line containing a single integer *m*, the number of points on the convex hull. Next output *m* lines, each describing a point on the convex hull, in counter-clockwise order around the hull. Each of these lines should contain the x-coordinate of the point, followed by a space, followed by the y-coordinate of the point. Start with the point on the hull whose x-coordinate is minimal. If there are multiple such points, start with the one whose y-coordinate is minimal.

Output for Sample Input

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

Ondřej Lhoták, Malcolm Sharpe

10066 - The Twin Towers

Time limit: 3.000 seconds

Once upon a time, in an ancient Empire, there were two towers of dissimilar shapes in two different cities. The towers were built by putting circular tiles one upon another. Each of the tiles was of the same height and had integral radius. It is no wonder that though the two towers were of dissimilar shape, they had many tiles in common.

However, more than thousand years after they were built, the Emperor ordered his architects to remove some of the tiles from the two towers so that they have exactly the same shape and size, and at the same time remain as high as possible. The order of the tiles in the new towers must remain the same as they were in the original towers. The Emperor thought that, in this way the two towers might be able to stand as the symbol of harmony and equality between the two cities. He decided to name them the *Twin Towers*.

Now, about two thousand years later, you are challenged with an even simpler problem: given the descriptions of two dissimilar towers you are asked only to find out the number of tiles in the highest twin towers that can be built from them.

Input

The input file consists of several data blocks. Each data block describes a pair of towers.

The first line of a data block contains two integers N1 and N2 (1 <= N1, N2 <= 100) indicating the number of tiles respectively in the two towers. The next line contains N1 positive integers giving the radii of the tiles (from top to bottom) in the first tower. Then follows another line containing N2 integers giving the radii of the tiles (from top to bottom) in the second tower.

The input file terminates with two zeros for N1 and N2.

Output

For each pair of towers in the input first output the twin tower number followed by the number of tiles (in one tower) in the highest possible twin towers that can be built from them. Print a blank line after the output of each data set.

Sample Input

```
7 6
20 15 10 15 25 20 15
15 25 10 20 15 20
8 9
10 20 20 10 20 10 20 10
20 10 20 10 10 20 10 10 20
0 0
```

Sample Output

```
Twin Towers #1
Number of Tiles : 4

Twin Towers #2
Number of Tiles : 6
```

111 - History Grading

Time limit: 3.000 seconds

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 $1 \leq q \leq n$ 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
5
10
9
```