A - I Can Guess the Data Structure!

Source file name: `guess.py`
Time limit: 1 second

There is a bag-like data structure, supporting two types of commands:

`1 x` throws element *x* into the bag, and

`2` takes out an element from the bag.

Given a sequence of operations with return values, you're going to guess the data structure. It is a stack (last-in, first-out), a queue (first-in, first-out), a priority-queue (always take out larger elements first), or something else that you can hardly imagine!

Input

There are several test cases. Each test case begins with a line containing a single integer n ($1 \leq n \leq 1000$). Each of the next n lines is either a type-1 command or an integer 2 followed by an integer x, meaning that after executing a type-2 command, we get an element x *without* error. The value of x is always a positive integer not larger than 100. The input is terminated by end-of-input.

The input must be read from standard input.

Output

For each test case, output one of the following:

`stack` if it is definitely a stack;

`queue` if it is definitely a queue;

`priority queue` if it is definitely a priority queue;

`impossible` if it is none of the above; and

`not sure` if it can be more than one of the data structures mentioned above.

The output must be written to standard output.

Sample Input	**Sample Output**

6
1 1
1 2
1 3
2 1
2 2
2 3
6
1 1
1 2
1 3
2 3
2 2
2 1
2
1 1
2 2
4
1 2
1 1
2 1
2 2
7
1 2
1 5
1 1
1 3
2 5
1 4
2 4
	queue
	not sure
	impossible
	stack
	priority queue

B - Rings and Glue

Source file name: `glue.py`
Time limit: 1 second

Little John is in big trouble. Playing with his different-sized (and colored!) rings and glue seemed such a good idea. However, the rings now lay on the floor, glued together with some-thing that will definitely not come off with water. Surprisingly enough, it seems like no rings are actually glued to the floor, only to other rings. How about that!

You must help Little John to pick the rings off the floor before his mom comes home from work. Since the glue is dry by now, it seems like an easy enough task. This is not the case. Little John is an irrational kid of numbers, and so he has decided to pick up the largest component (most rings) of glued-together rings first. It is the number of rings in this largest component you are asked to find. Two rings are glued together if and only if they overlap at some point but no rings will ever overlap in only a single point. All rings are of the doughnut kind (with a hole in them). They can however, according to Little John, be considered "infinitely thin".

Input

Input consists of several test cases. Each test case starts with the number of rings n, where $0 < n < 100$. After that, n rows follow, each containing a ring's physical attributes. That is, 3 floating point numbers separated by blanks each describing the x coordinate and y coordinate for its center, and its radius (which is non-negative).

Input ends with a single row with the integer -1.

The input must be read from standard input.

Output

Output consists of as many grammatically correct answers as there were problems, each answer, on a separate line, being

```
The largest component contains X ring(s).
```

where X is the number of rings in the largest component.

The output must be written to standard output.

Sample Input

```
 4
 0.0 0.0 1.0
-1.5 -1.5 0.5
 1.5 1.5 0.5
-2.0 2.0 3.5
 3
 3.0 2.0 2.0
 0.0 -0.5 1.0
 0.0 0.0 2.0
 1
 0.0 0.0 0.0
-1
```

Sample Output

```
The largest component contains 4 rings.
The largest component contains 2 rings.
The largest component contains 1 ring.
```

C - Conformity

Source file name: `conformity.py`
Time limit: 1 second

Frosh commencing their studies at Waterloo have diverse interests, as evidenced by their desire to take various combinations of courses from among those available.

University administrators are uncomfortable with this situation, and therefore wish to offer a *conformity prize* to frosh who choose the most popular combination of courses. How many frosh will win the prize? The popularity of a combination is the number of frosh selecting exactly the same combination of courses. A combination of courses is considered most popular if no other combination has higher popularity.

Input

The input consists of several test cases followed by a line containing 0. Each test case begins with an integer $1 \leq n \leq 10000$, the number of frosh. For each frosh, a line follows containing the course numbers of five distinct courses selected by the frosh. Each course number is an integer between 100 and 499.

The input must be read from standard input.

Output

For each line of input, you should output a single line giving the total number of students taking some combination of courses that is most popular.

The output must be written to standard output.

Sample Input	**Sample Output**

`3`
`100 101 102 103 488`
`100 200 300 101 102`
`103 102 101 488 100`
`3`
`200 202 204 206 208`
`123 234 345 456 321`
`100 200 300 400 444`
`0`
	`2`
	`3`

D - CD

Source file name: `cd.py`
Time limit: 1 second

Jack and Jill have decided to sell some of their Compact Discs, while they still have some value. They have decided to sell one of each of the CD titles that they both own. How many CDs can Jack and Jill sell?

Neither Jack nor Jill owns more than one copy of each CD.

Input

The input consists of a sequence of test cases. The first line of each test case contains two non-negative integers N and M, each at most one million, specifying the number of CDs owned by Jack and by Jill, respectively. This line is followed by N lines listing the catalog numbers of the CDs owned by Jack in increasing order, and M more lines listing the catalog numbers of the CDs owned by Jill in increasing order. Each catalog number is a positive integer no greater than one billion.

The input is terminated by a line containing two zeros. This last line is not a test case and should not be processed.

The input must be read from standard input.

Output

For each test case, output a line containing one integer, the number of CDs that Jack and Jill both own.

The output must be written to standard output.

Sample Input	**Sample Output**

`3 3`
`1`
`2`
`3`
`1`
`2`
`4`
`0 0`
	`2`