Problem 10008

What's Cryptanalysis?

Cryptanalysis is the process of breaking someone else's cryptographic writing. This sometimes involves some kind of statistical analysis of a passage of (encrypted) text. Your task is to write a program which performs a simple analysis of a given text.

Input

The first line of input contains a single positive decimal integer *n*. This is the number of lines which follow in the input. The next *n* lines will contain zero or more characters (possibly including whitespace). This is the text which must be analyzed.

Output

Each line of output contains a single uppercase letter, followed by a single space, then followed by a positive decimal integer. The integer indicates how many times the corresponding letter appears in the input text. Upper and lower case letters in the input are to be considered the same. No other characters must be counted. The output must be sorted in descending count order; that is, the most frequent letter is on the first output line, and the last line of output indicates the least frequent letter. If two letters have the same frequency, then the letter which comes first in the alphabet must appear first in the output. If a letter does not appear in the text, then that letter must not appear in the output.

Sample Input

```
3
This is a test.
Count me 1 2 3 4 5.
Wow!!!!  Is this question easy?
```

Sample Output

```
S 7
T 6
I 5
E 4
O 3
A 2
H 2
N 2
U 2
W 2
C 1
M 1
Q 1
Y 1
```

Problem 10162

Problem B.Last Digit

Background

Give you a integer number N ($1<=n<=2*10^{100}$). Please compute

$$S=1^1+2^2+3^3+\ldots+N^N$$

Give the last digit of S to me.

Input

Input file consists of several Ns, each N a line. It is ended with N=0.

Output

For each N give a line containing only one digit, which is the last digit of S.

Sample Input

```
1
2
3
0
```

Sample Output

```
1
5
2
```

Problem 10192

Vacation

The Problem

You are planning to take some rest and to go out on vacation, but you really don't know which cities you should visit. So, you ask your parents for help. Your mother says *"My son, you MUST visit Paris, Madrid, Lisboa and London. But it's only fun in this order."* Then your father says: *"Son, if you're planning to travel, go first to Paris, then to Lisboa, then to London and then, at last, go to Madrid. I know what I'm talking about."*

Now you're a bit confused, as you didn't expected this situation. You're afraid that you'll hurt your mother if you follow your father's suggestion. But you're also afraid to hurt your father if you follow you mother's suggestion. But it can get worse, because you can hurt both of them if you simply ignore their suggestions!

Thus, you decide that you'll try to follow their suggestions in the better way that you can. So, you realize that the "Paris-Lisboa-London" order is the one which better satisfies both your mother and your father. Afterwards you can say that you could not visit Madrid, even though you would've liked it very much.

If your father have suggested the "London-Paris-Lisboa-Madrid" order, then you would have two orders, "Paris-Lisboa" and "Paris-Madrid", that would better satisfy both of your parent's suggestions. In this case, you could only visit 2 cities.

You want to avoid problems like this one in the future. And what if their travel suggestions were bigger? Probably you would not find the better way very easy. So, you decided to write a program to help you in this task. You'll represent each city by one character, using uppercase letters, lowercase letters, digits and the space. Thus, you can have at most 63 different cities to visit. But it's possible that you'll visit some city more than once.

If you represent Paris with "a", Madrid with "b", Lisboa with "c" and London with "d", then your mother's suggestion would be "abcd" and you father's suggestion would be "acdb" (or "dacb", in the second example).

The program will read two travel sequences and it must answer how many cities you can travel to such that you'll satisfy both of your parents and it's maximum.

The Input

The input will consist on an arbitrary number of city sequence pairs. The end of input occurs when the first sequence starts with an "#"character (without the quotes). Your program should not process this case. Each travel sequence will be on a line alone and will be formed by legal characters (as defined above). All travel sequences will appear in a single line and will have at most 100 cities.

The Output

For each sequence pair, you must print the following message in a line alone:

```
Case #d: you can visit at most K cities.
```

Where d stands for the test case number (starting from 1) and K is the maximum number of cities you can visit such that you'll satisfy both you father's suggestion and you mother's suggestion.

Sample Input

```
abcd
acdb
abcd
dacb
#
```

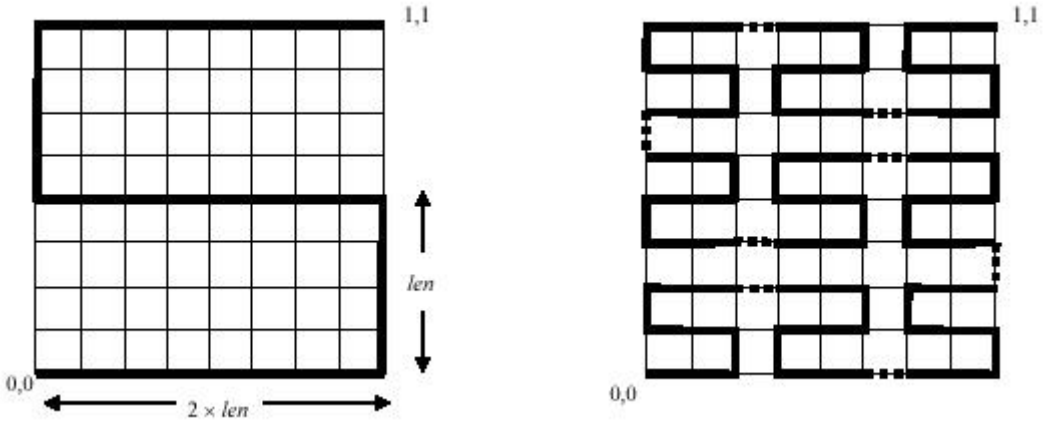
Sample Output

```
Case #1: you can visit at most 3 cities.
Case #2: you can visit at most 2 cities.
```

Problem 1020

The latest research in reconfigurable multiprocessor chips focuses on the use of a single bus that winds around the chip. Processor components, which can be anywhere on the chip, are attached to *connecting points* on the bus so that they can communicate with each other.

Some research involves bus layout that uses recursively-defined ‘‘SZ" curves, also known as ‘‘S-shaped Peano curves." Two examples of these curves are shown below. Each curve is drawn on the unit square. The order-1 curve, shown on the left, approximates the letter ‘‘S" and consists of line segments connecting the points (0,0), (1,0), (1,0.5), (0,0.5), (0,1), and (1,1) in order. Each horizontal line in an ‘‘S" or ‘‘Z" curve is twice as long as each vertical line. For the order-1 curve, the length of a vertical line, *len*, is 0.5.



The order-2 curve, shown on the right, contains 9 smaller copies of the order-1 curve (4 of which are reversed left to right to yield ‘‘Z" curves). These copies are connected by line segments of length *len*, shown as dotted lines. Since the width and height of the order-2 curve is 8×*len*, and the curve is drawn on the unit square, *len* = 0.125 for the order-2 curve.

The order-3 curve contains 9 smaller copies of the order-2 curve (with 4 reversed left to right), connected by line segments, as described for the order-2 curve. Higher order curves are drawn in a similar manner. The *connecting points* to which processor components attach are evenly spaced every *len* units along the bus. The first connecting point is at (0,0) and the last is at (1,1). There are 9^k connecting points along the order-*k* curve, and the total bus length is $(9^k - 1)\times len$ units.

You must write a program to determine the total distance that signals must travel between two processor components. Each component’s coordinates are given as an *x*, *y* pair, $0 \leq x \leq 1$ and $0 \leq y \leq 1$, where *x* is the distance from the left side of the chip, and *y* is the distance from the lower edge of the chip. Each component is attached to the closest connecting point by a straight line. If multiple connecting points are equidistant from a component, the one with the smallest *x* coordinate and smallest *y* coordinate is used. The total distance a signal must travel between two components is the sum of the length of the lines connecting the components to the bus, and the length of the bus between the two connecting points. For example, the distance between components located at (0.5, 0.25) and (1.0, 0.875) on a chip using the order-1 curve is 3.8750 units.

Input

The input contains several cases. For each case, the input consists of an integer that gives the order of the

SZ curve used as the bus (no larger than 8), and then four real numbers x_1, y_1, x_2, y_2 that give the coordinates of the processor components to be connected. While each processor component should actually be in a unique location not on the bus, your program must correctly handle all possible locations.

The last case in the input is followed by a single zero.

Output

For each case, display the case number (starting with 1 for the first case) and the distance between the processor components when they are connected as described. Display the distance with 4 digits to the right of the decimal point.

Use the same format as that shown in the sample output shown below. Leave a blank line between the output lines for consecutive cases.

Sample Input

```
1 0.5 .25 1 .875
1 0 0 1 1
2 .3 .3 .7 .7
2 0 0 1 1
0
```

Sample Output

```
Case 1.  Distance is 3.8750

Case 2.  Distance is 4.0000

Case 3.  Distance is 8.1414

Case 4.  Distance is 10.0000
```

Problem 10818: Dora Trip

Nobita is in great trouble. Today he failed to hand in his homework again, so he was heavily punished at school. Learning that, his mother gets furious, and therefore assigns him many tasks to do - to buy vegetables at the market, to collect a parcel at the post office and a lot more. Nobita certainly does not want to see his teacher on his way, nor would he like to meet Jyian, the tough bully. As usual, he asks Doraemon for help.

"Oh no!" cried Doraemon. "My *everywhere door* is broken, and my *small propellers* have all run out of batteries..." Well, that means Nobita has got to go without Doraemon's magic tools. "Ah, I still have this. It may well be useful." From his 4th-dimensional pocket, Doraemon takes out a map of their living area. He then marks on it the places where Nobita has to visit by asterisks ('*****'), and where Jyian or his teacher may appear by crosses ('**X**'). Now Nobita's job is simple - he has to find the shortest route, through which he would not visit any of the 'crosses', and he could finish the maximum number of the jobs (if not all) given by mum. What he needs is just a computer program that works out the path...

Imagine that you are Nobita. Write the program.

Input

The input file contains no more than **20** test cases. The details of each set are given as follows:

The first line of each case contains two integers ***r*** and ***c*** (1 <= ***r***, ***c*** <= 20), which are the number of rows and columns of the map respectively. The next ***r*** lines, each with ***c*** characters, give the map itself. For each character, a space `` " stands for an open space; a hash mark ``**#**" stands for an obstructing wall; the capital letter ``**S**" stands for the position of Nobita's house, which is where his journey is to start and end; the capital letter ``**X**" stands for a dangerous place; and an asterisk ``**\***" stands for a place he has to visit. The perimeter of the map is always closed, i.e., there is no way to get out from the coordinate of the ``**S**". The number of places that Nobita has to visit is at most **10**.

The input file is terminated by a null case where ***r*** **= *c* = 0**. This case should not be processed.

Output

For each test case, if Nobita cannot visit any target places at all, just print the line "**Stay home!**". Otherwise, your program should output the lexicographically smallest shortest path so that the number of target places that Nobita visits is maximized. Use the letters '**N**', '**S**', '**E**' and '**W**' to denote north, south, east and west respectively. Note that by 'north' we mean facing upwards. You can be sure that the length of a correct output path will never exceed 200.

Sample Input

```
5 5
```

Sample Output

```
WWSSEEWWNNEE
```

```
#####
#  S#
# XX#
#  *#
#####
5 5
#####
#* X#
###X#
#S *#
#####
5 5
#####
#S X#
#  X#
# #*#
#####
0 0
```

```
EEWW
Stay home!
```

Problem 11450

Wedding Shopping

Background

One of our best friends is getting married and we all are nervous because he is the first of us who is doing something similar. In fact, we have never assisted to a wedding, so we have no clothes or accessories, and to solve the problem we are going to a famous department store of our city to buy all we need: a shirt, a belt, some shoes, a tie, etcetera.

The Problem

We are offered different models for each class of garment (for example, three shirts, two belts, four shoes, ..). We have to buy one model of each class of garment, and just one.

As our budget is limited, we cannot spend more money than it, but we want to spend the maximum possible. It's possible that we cannot buy one model of each class of garment due to the short amount of money we have.

The Input

The first line of the input contains an integer, N, indicating the number of test cases. For each test case, some lines appear, the first one contains two integers, M and C, separated by blanks ($1<=M<=200$, and $1<=C<=20$), where M is the available amount of money and C is the number of garments you have to buy. Following this line, there are C lines, each one with some integers separated by blanks; in each of these lines the first integer, K ($1<=K<=20$), indicates the number of different models for each garment and it is followed by K integers indicating the price of each model of that garment.

The Output

For each test case, the output should consist of one integer indicating the maximum amount of money necessary to buy one element of each garment without exceeding the initial amount of money. If there is no solution, you must print "`no solution`".

Sample Input

```
100 4
3 8 6 4
2 5 10
4 1 3 3 7
4 50 14 23 8
20 3
3 4 6 8
2 5 10
4 1 3 5 5
5 3
3 6 4 8
2 10 6
4 7 3 1 7
```

Sample Output

```
75
19
no solution
```

Problem 11985: Prime Independence

A set of integers is called prime independent if none of its member is a prime multiple of another member. An integer **a** is said to be a **prime multiple** of **b** if,

a = b × k (where **k** is a prime [1])

So, **6** is a prime multiple of **2**, but **8** is not. And for example, **{2, 8, 17}** is prime independent but **{2, 8, 16}** or **{3, 6}** are not.

Now, given a set of distinct positive integers, calculate the largest prime independent subset.

Input

Input starts with an integer **T (≤ 25)**, denoting the number of test cases.

Each case starts with an integer **N (1 ≤ N ≤ 40000)** denoting the size of the set. Next line contains **N** integers separated by a single space. Each of these **N** integers are distinct and between **1** and **500000** inclusive.

Output

For each case, print the case number and the size of the largest prime independent subset.

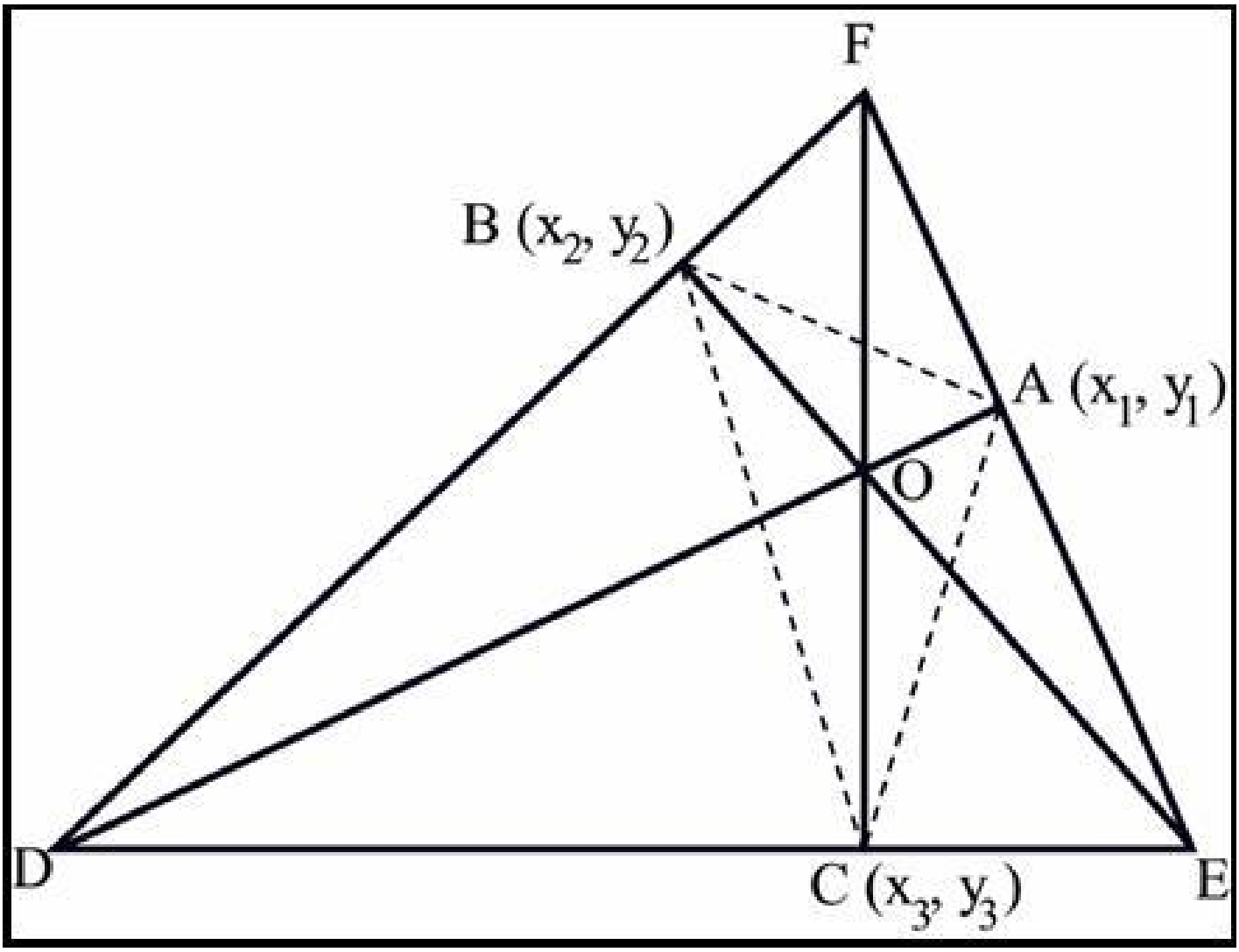
Sample Input	Output for Sample Input
3 5 2 4 8 16 32 5 2 3 4 6 9 3 1 2 3	Case 1: 3 Case 2: 3 Case 3: 2

Notes

1. An integer is said to be a prime if it's divisible by exactly two distinct integers. First few prime numbers are **2, 3, 5, 7, 11, 13, ...**

Problem 12073

If DEF is an acute triangle and DA, EB and FC are its three heights on EF, DF and DE respectively, then the triangle ABC is called the altitude triangle of triangle DEF. It is well known that DA, EB and FC are concurrent and let us assume that their common point of intersection is O. So point O is called the orthocenter of triangle DEF. It can be proved that O is the in center of triangle ABC. In this problem you will be given the altitude triangle ABC and your job is to find out the corresponding acute triangle DEF.



Input

The input file contains at most 2000 lines of input. Each line contains six integers x_1 , y_1 , x_2 , y_2 and x_3 , y_3 . These six integers denote an altitude triangle with vertex A (x_1, y_1) , B (x_2, y_2) and C (x_3, y_3) respectively. Input is terminated by a case where all six integers are zero. The points A, B and C will not be collinear.

Output

For each line of input produce four lines of outputs. The description of these four lines is given below:

The first line contains the serial of output. Each of the next three lines contains two floating-point

numbers, which are actually the coordinate of D, E and F respectively. Note that for a given altitude triangle ABC, there can be four possible triangles DEF. But you are requested only to find the one that is acute. Also note that judge data will be such that precision errors should not occur if you use double precision floating-point numbers. Absolute values of none of the output numbers will be greater than 100000 and all the numbers should have three digits after the decimal point.

Sample Input

```
682 1369 3981 1233 4333 4583
4131 734 1249 4705 2815 475
2815 475 4131 734 1249 4705
0 0 0 0 0 0
```

Sample Output

```
Case 1:
6539.582 3443.107
-1528.155 7610.801
1491.578 -917.367
Case 2:
-1810.802 3068.269
3810.093 -82.858
6872.845 7713.274
Case 3:
6872.845 7713.274
-1810.802 3068.269
3810.093 -82.858
```

Problem 1220

Dear Contestant,

I'm going to have a party at my villa at Hali-Bula to celebrate my retirement from BCM. I wish I could invite all my co-workers, but imagine how an employee can enjoy a party when he finds his boss among the guests! So, I decide not to invite both an employee and his/her boss. The organizational hierarchy at BCM is such that nobody has more than one boss, and there is one and only one employee with no boss at all (the Big Boss)! Can I ask you to please write a program to determine the maximum number of guests so that no employee is invited when his/her boss is invited too? I've attached the list of employees and the organizational hierarchy of BCM.

Best,
-Brian Bennett

P.S. I would be very grateful if your program can indicate whether the list of people is uniquely determined if I choose to invite the maximum number of guests with that condition.

Input

The input consists of multiple test cases. Each test case is started with a line containing an integer n ($1 \leq n \leq 200$) , the number of BCM employees. The next line contains the name of the Big Boss only. Each of the following n - 1 lines contains the name of an employee together with the name of his/her boss. All names are strings of at least one and at most 100 letters and are separated by blanks. The last line of each test case contains a single `0`.

Output

For each test case, write a single line containing a number indicating the maximum number of guests that can be invited according to the required condition, and a word `Yes` or `No`, depending on whether the list of guests is unique in that case.

Sample Input

```
6
Jason
Jack Jason
Joe Jack
Jill Jason
John Jack
Jim Jill
2
Ming
Cho Ming
0
```

Sample Output

```
4 Yes
1 No
```

Problem 477

Points in Figures: Rectangles and Circles

Given a list of figures (rectangles and circles) and a list of points in the *x*-*y* plane, determine for each point which figures (if any) contain the point.

Input

There will be *n*(≤ 10) figures descriptions, one per line. The first character will designate the type of figure (``r", ``c" for rectangle or circle, respectively). This character will be followed by values which describe that figure.

- For a rectangle, there will be four real values designating the *x*-*y* coordinates of the upper left and lower right corners.
- For a circle, there will be three real values, designating the *x*-*y* coordinates of the center and the radius.

The end of the list will be signalled by a line containing an asterisk in column one.

The remaining lines will contain the *x*-*y* coordinates, one per line, of the points to be tested. The end of this list will be indicated by a point with coordinates 9999.9 9999.9; these values should not be included in the output.

Points coinciding with a figure border are not considered inside.

Output

For each point to be tested, write a message of the form:

```
Point i is contained in figure j
```

for each figure that contains that point. If the point is not contained in any figure, write a message of the form:

```
Point i is not contained in any figure
```

Points and figures should be numbered in the order in which they appear in the input.

Sample Input

```
r 8.5 17.0 25.5 -8.5
c 20.2 7.3 5.8
r 0.0 10.3 5.5 0.0
c -5.0 -5.0 3.7
r 2.5 12.5 12.5 2.5
c 5.0 15.0 7.2
*
2.0 2.0
4.7 5.3
6.9 11.2
20.0 20.0
17.6 3.2
-5.2 -7.8
```

```
9999.9 9999.9
```

Sample Output

```
Point 1 is contained in figure 3
Point 2 is contained in figure 3
Point 2 is contained in figure 5
Point 3 is contained in figure 5
Point 3 is contained in figure 6
Point 4 is not contained in any figure
Point 5 is contained in figure 1
Point 5 is contained in figure 2
Point 6 is contained in figure 4
```

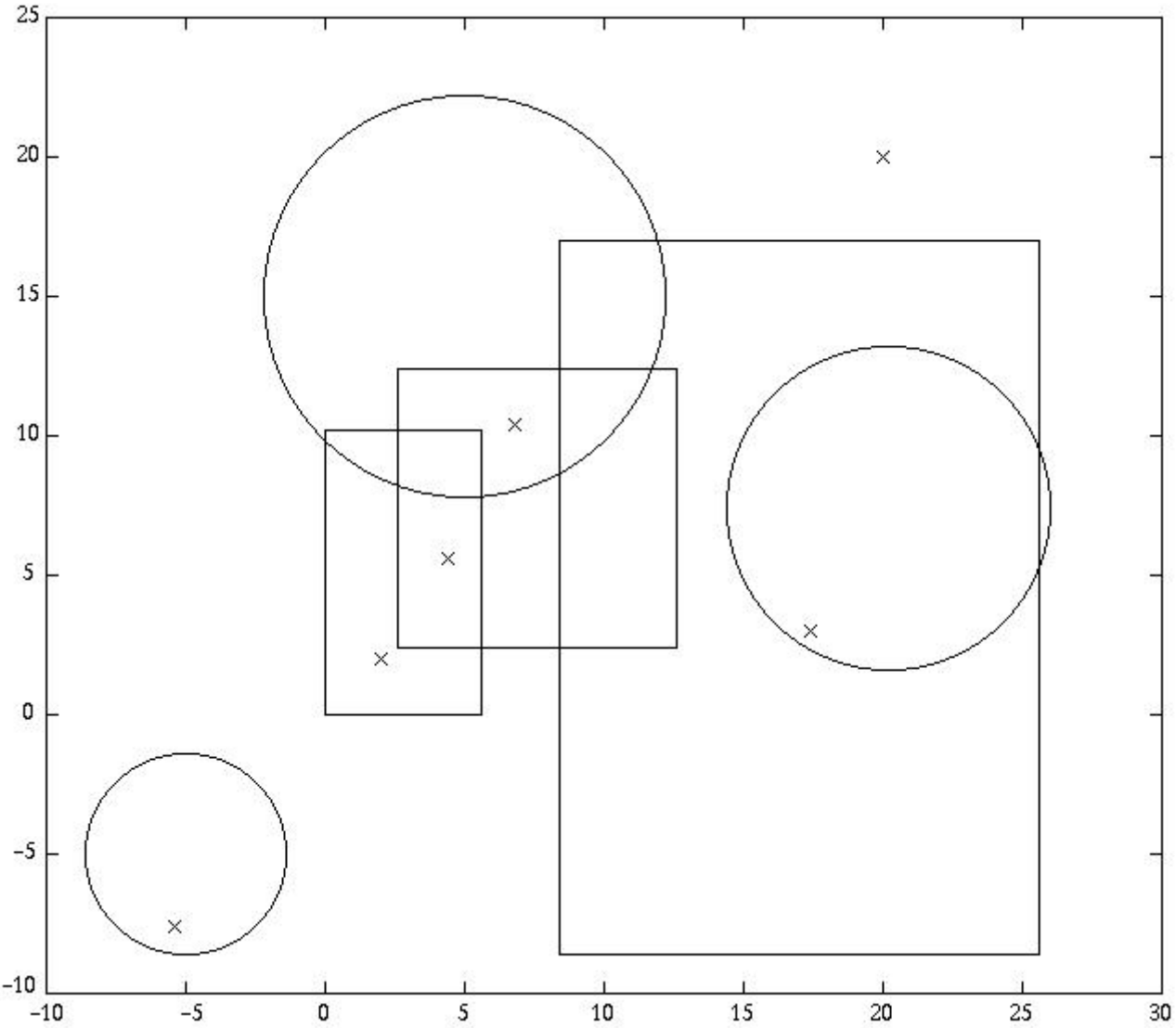


Diagrama of sample input figures and data points