Problem A: International Terrorists

Interpol, ever vigilant in the struggle against international terrorists, has just passed on the following two vital scraps of information about a terrorist about to land at your airport:

- There are three letters the same in the terrorist's surname; and
- The terrorist's passport number has two 6s or two 7s which are not adjacent.

The airlines have given you a list of the names and passport numbers of people soon to arrive in the country. You must write a program to print out this data for any who could be the terrorist.

INPUT

The input to your program is a list of arriving passengers' names and passport numbers. The first 30 characters of each line is the passenger's name, and the rest is the passport number. The name is given surname first, followed by a comma, then the other parts of the name. The surname never contains a comma but may contain characters other than letters. The surname could possibly be empty, in which case the line will start with a comma. Exactly 30 characters are always given for the name; if the whole name is less than 30 characters long, the end is padded with spaces. The passport numbers have very different formats depending on the country that issued them, but they are at most 30 characters long and never include spaces. The input will be terminated by a line consisting of a single `#`.

OUTPUT

Write out the lines from the input that, according to Interpol, belong to passengers who could be the terrorist.

SAMPLE INPUT

```
Mmd abd el Qiro, Saleem MuzrobAC67673BC
Lennon, Michael James John    R626584
Bibit Gobet, Fromik Danxz     6X7Y66J777P56
Zzzzqqq, Xxxxx Yyyyy          666
Pipit Gopet, Fromik Danxz     4X5Y66J77ZP54
,Zzzzqqq Xxxxx Yyyyy          777
#
```

SAMPLE OUTPUT

```
Lennon, Michael James John    R626584
Bibit Gobet, Fromik Danxz     6X7Y66J777P56
Zzzzqqq, Xxxxx Yyyyy          666
```

Problem B: Spirals

When n is an odd integer, the numbers from 1 to n^2 can be arranged in a spiral within an n by n matrix, like this:

13	12	11	10	25
14	3	2	9	24
15	4	1	8	23
16	5	6	7	22
17	18	19	20	21

Each number, m, in the spiral can be described by its position in the matrix. For example, 18 is at position $(5, 2)$ because it is in the fifth row and the second column.

Your task is to write a program that, given n and m, works out the position of m in the n by n matrix.

INPUT

Each line of input contains two numbers, n and m, separated by a space. n will always be an odd integer less than $40,000$. m is an integer from 1 to n^2. The end of input is indicated by a line with two zeros.

OUTPUT

For each pair of numbers n and m in the input, print the row and column of m in the $n \times n$ matrix, separated by a space.

SAMPLE INPUT

```
5 18
5 14
1 1
101 10001
1001 1002000
0 0
```

SAMPLE OUTPUT

```
5 2
2 1
1 1
101 1
2 1001
```

Problem C: KWIC Index

A popular automatically-generated index for technical documents is the KWIC (Key Word In Context) index. This is created by analysing a document, extracting all sentences which contain keywords, then rewriting these sentences so the keyword comes first.

For example, if all words in the sentence are potential keywords, there are 7 KWIC index entries for the phrase

```
Programming competitions are a lot of fun
```

They are formed by adding a comma to the end of the sentence, capitalising the keyword and rearranging to move the keyword to the start. This is repeated for all keywords appearing in the index, and then the list of KWIC entries is sorted in alphabetical order. So the complete KWIC index for the phrase above is:

```
A lot of fun, Programming competitions are
ARE a lot of fun, Programming competitions
COMPETITIONS are a lot of fun, Programming
FUN, Programming competitions are a lot of
LOT of fun, Programming competitions are a
OF fun, Programming competitions are a lot
PROGRAMMING competitions are a lot of fun,
```

Then if you are asked for the third KWIC entry for this phrase, the answer would be

```
COMPETITIONS are a lot of fun, Programming
```

INPUT

The input consists of a number of phrases for which a specified KWIC entry must be found. Preceding each phrase is a line containing a number n, which is between 1 and the number of words in the phrase. The phrase has from 1 to 50 words, each at most 20 letters long and separated by single spaces. It will be given on at most four lines of up to 60 characters long, with no space characters at the start and end of the lines. The end of a phrase is marked by a line consisting of a single `#`. The input is terminated by a line containing only a zero.

OUTPUT

For each number-phrase combination in the input, print the first 60 characters of the nth entry in the KWIC index for the phrase.

SAMPLE INPUT

```
3
Programming competitions are a lot of fun
#
16
A popular automatically generated index for technical
documents is the KWIC (Key Word In
Context) index
#
0
```

SAMPLE OUTPUT

```
COMPETITIONS are a lot of fun, Programming
WORD In Context) index, A popular automatically generated in
```

Problem D: Near-Perfection

The Greeks named a number *perfect* if it was the sum of its factors excluding itself. For example, 6 is perfect because its factors are $\{1, 2, 3, 6\}$, and $1 + 2 + 3 = 6$. 28 is also perfect.

Near-perfect numbers are those which are nearly equal to the sum of their factors. To be more precise, an x-perfect number is one which differs from the sum of its factors (excluding itself) by at most x. For example, the factors of 20 excluding itself sum to $1 + 2 + 4 + 5 + 10 = 22$, which differs from 20 by 2. So 20 is not 0-perfect nor 1-perfect, but is x-perfect for $x \geq 2$. Similarly, it is easy to see that 9 is x-perfect for any $x \geq 5$.

For this problem, you will be given pairs of numbers a and b and will be asked to calculate the largest a-perfect number which does not exceed b. Both a and b can be from 0 to 10,000.

INPUT

Each line of input contains the pair of numbers a and b, separated by a space. The test data could be up to a thousand lines long. The end of the input is indicated by a line containing two zeros.

OUTPUT

For each line of input, print the largest a-perfect number less than or equal to b. If there are no a-perfect numbers within this range, print 0.

SAMPLE INPUT

```
12 17
17 100
0 1000
0 0
```

SAMPLE OUTPUT

```
16
100
496
```

Problem E: Coloured Convoys

A convoy of coloured cars is travelling along a highway for the annual Graphic Designers' Mystery Holiday. The holiday organiser is not satisfied by the convoy's arrangement of colours, so she decides to rearrange them into an order that she finds more pleasing.

For simplicity, describe the convoy by a string such as `rrg-bvoib`. The letters show the cars' colours, with the first letter representing the car at the front of the convoy. The cars have the colours of the rainbow, so the letters come from the set `roygbiv`. The convoy's description may contain a `-`, which indicates a gap one car long in the middle of the convoy. At any time, there is at most one gap anywhere in the convoy.

The convoy's colour pattern is rearranged by instructing one car at a time to overtake a few cars then drop back into line. Since overtaking is a dangerous manoeuvre, the maximum number of cars overtaken is limited to some integer n, where $n \geq 1$. Then the allowable overtaking moves are

- Move forward one space into a gap, so that `r-bgr` becomes `rb-gr` and `rbgr` becomes `r-bgr`.
- Overtake one space, so that `r-bgr` becomes `rgb-r` and `rbgr` becomes `br-gr`.
- If $n \geq 2$, overtake two spaces, so that `r-bgr` becomes `rrbg` and `rbgr` becomes `grb-r` (note that `rbgr` cannot become `b-r-gr` because the convoy would then have two gaps).
- And so on, for overtaking $3, \ldots, n-1, n$ spaces at once.

The aim is to find the minimum number of overtaking manoeuvres that are required to transform the convoy from an initial to a final order (both without any gaps), given n, the maximum number of positions able to be overtaken at once.

INPUT

The input consists of descriptions of a number of convoys, each on three lines. The first line specifies n, the maximum number of cars that may be overtaken at once. The second and third lines are the strings representing the initial and final order of cars in the convoy, where the first character on the line is at the front of the convoy. The maximum length of the convoy is 15 vehicles. The end of the input is indicated by a line for n equal to 0.

OUTPUT

For each convoy, print the minimum number of overtaking manoeuvres needed to rearrange the convoy.

SAMPLE INPUT

```
2
rbrb
bbrr
2
rbrb
rrbb
0
```

SAMPLE OUTPUT

```
2
3
```

Problem F: Dates

Since the Earth takes about 365.24 days to orbit the Sun, our calendar year is normally 365 days long, but lengthens to 366 days when leap years occur. In many calendar systems, the year is broken into months, so any day can be represented as $d/m/y$, where d is the day of the month, m the month of the year and y the year since the calendar system began.

The Gregorian calendar, which we use, divides the year into 12 months with 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30 and 31 days respectively. The second month is extended to 29 days in a leap year. The pattern of leap years follows a 400-year cycle with 97 leap years. These occur when the number of the year is divisible by 4 and not by 100, or is divisible by 400 (so that 1900 was not a leap year, but 2000 will be).

The Persian calendar, probably the one which best matches the Earth's rotation rate, has months with 31, 31, 31, 31, 31, 31, 30, 30, 30, 30, 30 and 29 days respectively. The extra day in a leap year is added to the twelfth month, giving it 30 days. The pattern of leap years follows a 2,820-year cycle with 683 leap years. These 2,820 years are divided into 21 subcycles of 128 years followed by a cycle of 132 years. Each 128-year subcycle is divided into sub-subcycles of lengths 29, 33, 33, and 33 years, while the 132-year subcycle has sub-subcycles of 29, 33, 33 and 37 years. Within each sub-subcycle, the years $5, 9, 13, \ldots$ are leap years. Thus, counting from the beginning of the calendar, the leap years were years

$$5, 9, \ldots, 29, 34, 38, \ldots, 62, 67, 71, \ldots, 95, 100, 104, \ldots, 128, 133, \ldots$$

The Persian calendar started on the day that the Gregorian calendar called 19th March 622. Thus 1/1/1 in the Persian calendar is the same day as 19/3/622 in the Gregorian calendar.

For this problem you will be given four numbers d_g, m_g, d_p and m_p that specify a day and a month in the Gregorian and Persian calendars. You must calculate the earliest years y_g and y_p for the two calendar systems such that $d_g/m_g/y_g$ and $d_p/m_p/y_p$ refer to the same day in the Gregorian and Persian calendars respectively. Both y_g and y_p must be positive numbers.

INPUT

Each line of input has the four numbers d_g, m_g, d_p and m_p, separated by a space character. The end of input is indicated by a line with four zeros.

OUTPUT

For each set of Gregorian and Persian day and month numbers, print the two years y_g and y_p separated by a single space.

SAMPLE INPUT

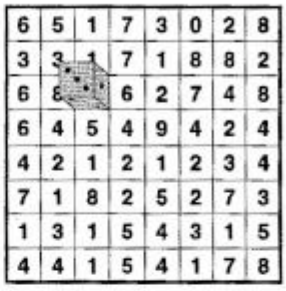
```
19 3 1 1
18 3 1 1
1 1 1 1
1 1 19 3
29 2 30 12
0 0 0 0
```

SAMPLE OUTPUT

```
622 1
624 3
248493 247872
514089 513468
54492 53870
```

Problem G: Roll'n'Score

Roll'n'Score is a new game of skill played by rolling a die around on a chessboard. The chessboard has the numbers 0 and 9 on two squares, and the numbers from 1 to 8 on the remaining 62 squares. The die is a normal die with six faces numbered 1 to 6. The relative position of three of the faces is shown in the diagram and the numbers on opposite faces always sum to 7.



To begin the game, the die is placed on the square numbered 0, on any of its faces, with its sides aligned with the sides of the chessboard. The aim is to find the lowest score possible while rolling the die from square to square around the board until it lands on the square with the number 9. Each time the die rests on a new square, the score increases by the product of the number of this new square with the number on the face of the die that touches it (thus players try to arrange for the die to land on the final square on the face numbered 1).

For example, for the chessboard in the picture, the lowest score is 31, which starts with the die on the 0 in the first row with 4 uppermost, 2 pointing to the left and 1 towards the top of the board. Rolling the die left once and down three times gives the score of

$$2 \times 3 + 6 \times 1 + 5 \times 2 + 1 \times 9 = 31$$

INPUT

The input consists of descriptions of a number of chessboards, as shown in the sample input. Each board is specified by exactly 8 lines, giving the numbers in each row, starting at the top of the board. Each line lists the 8 numbers in the squares from left to right, separated by spaces. The input is terminated by a line consisting of eight zeros.

OUTPUT

For each board in the input, print a line of the form

```
The lowest score is 31
```

showing the lowest score possible.

SAMPLE INPUT

```
6 5 1 7 3 0 2 8
3 3 1 7 1 8 8 2
6 8 8 6 2 7 4 8
6 4 5 4 9 4 2 4
4 2 1 2 1 2 3 4
7 1 8 2 5 2 7 3
1 3 1 5 4 3 1 5
4 4 1 5 4 1 7 8
1 1 1 1 1 1 1 0
2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7
9 8 8 8 8 8 8 8
0 0 0 0 0 0 0 0
```

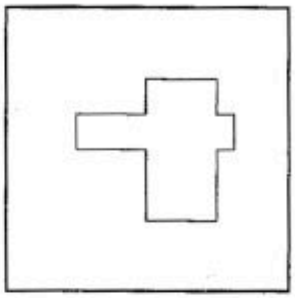
SAMPLE OUTPUT

```
Lowest score is 31
Lowest score is 130
```

Problem H: Hole Punch

A manufacturing firm specialises in cutting very accurately located holes into sheets of stainless steel. These holes have sides that are always parallel to the sheet's edges.

The holes used to be cut by hand, but the company has just purchased a press that punches out rectangular holes in the steel sheets. You have to write a program that takes a list of the vertices of a hole to be punched into a steel sheet and determines the minimum number of rectangular punches needed to form the same hole with the new press. For example, the hole shown in the picture could be formed by stamping out two rectangles.



INPUT

The input consists of descriptions of a number of holes. Each description is a list of the coordinates of the corners of the hole. There could be up to 1000 corners for each hole. Each coordinate is a pair of integers from 5 to 9995, separated by a space, giving the x and y coordinates of the corner. The coordinates for one hole will be listed on possibly several lines of length at most 80 characters, with up to 10 coordinate pairs on each line; two coordinate pairs are separated by a space. The list of corners described by the coordinate pairs starts with the corner which, among all corners with maximum y coordinate, has the maximum x coordinate, and then continues clockwise around the hole. The end of each hole description will be marked by a line containing two zeros. The end of the input will be marked by a hole description with no corners in it.

OUTPUT

For each hole description, print the smallest number of rectangles that can be stamped out to make that hole.

SAMPLE INPUT

```
80 110 80 90 100 90 100 70 80 70 80 50 100 50
100 30 80 30 80 10 60 10 60 30 40 30 40 50
60 50 60 70 40 70 40 90 60 90 60 110
0 0
0 0
```

SAMPLE OUTPUT

```
3
```