Problem A
Unreliable Messengers
Input: message.txt

The King of a little Kingdom on a little island in the Pacific Ocean frequently has childish ideas. One day he said, "You shall make use of a message relaying game when you inform me of something." In response to the King's statement, six servants were selected as messengers whose names were Mr. J, Miss C, Mr. E, Mr. A, Dr. P, and Mr. M. They had to relay a message to the next messenger until the message got to the King.

Messages addressed to the King consist of digits (`'0'`–`'9'`) and alphabet characters (`'a'`–`'z'`, `'A'`–`'Z'`). Capital and small letters are distinguished in messages. For example, "`ke3E9Aa`" is a message.

Contrary to King's expectations, he always received wrong messages, because each messenger changed messages a bit before passing them to the next messenger. Since it irritated the King, he told you who are the Minister of the Science and Technology Agency of the Kingdom, "We don't want such a wrong message any more. You shall develop software to correct it!" In response to the King's new statement, you analyzed the messengers' mistakes with all technologies in the Kingdom, and acquired the following features of mistakes of each messenger. A surprising point was that each messenger made the same mistake whenever relaying a message. The following facts were observed.

Mr. J rotates all characters of the message to the left by one. For example, he transforms "`aB23d`" to "`B23da`".

Miss C rotates all characters of the message to the right by one. For example, she transforms "`aB23d`" to "`daB23`".

Mr. E swaps the left half of the message with the right half. If the message has an odd number of characters, the middle one does not move. For example, he transforms "`e3ac`" to "`ace3`", and "`aB23d`" to "`3d2aB`".

Mr. A reverses the message. For example, he transforms "`aB23d`" to "`d32Ba`".

Dr. P increments by one all the digits in the message. If a digit is `'9'`, it becomes `'0'`. The alphabet characters do not change. For example, he transforms "`aB23d`" to "`aB34d`", and "`e9ac`" to "`e0ac`".

Mr. M decrements by one all the digits in the message. If a digit is `'0'`, it becomes `'9'`. The alphabet characters do not change. For example, he transforms "`aB23d`" to "`aB12d`", and "`e0ac`" to "`e9ac`".

The software you must develop is to infer the original message from the final message, given the order of the messengers. For example, if the order of the messengers is A→J→M→P and the message given to the King is “`aB23d`”, what is the original message? According to the features of the messengers’ mistakes, the sequence leading to the final message is

$$\text{“}\texttt{32Bad}\text{”} \xrightarrow{\text{A}} \text{“}\texttt{daB23}\text{”} \xrightarrow{\text{J}} \text{“}\texttt{aB23d}\text{”} \xrightarrow{\text{M}} \text{“}\texttt{aB12d}\text{”} \xrightarrow{\text{P}} \text{“}\texttt{aB23d}\text{”}.$$

As a result, the original message should be “`32Bad`”.

Input

The input format is as follows.

n
The order of messengers
The message given to the King
⋮
The order of messengers
The message given to the King

The first line of the input contains a positive integer n, which denotes the number of data sets. Each data set is a pair of the order of messengers and the message given to the King. The number of messengers relaying a message is between 1 and 6 inclusive. The same person may not appear more than once in the order of messengers. The length of a message is between 1 and 25 inclusive.

Output

The inferred messages are printed each on a separate line.

Sample Input

```
5
AJMP
aB23d
E
86AE
AM
6
JPEM
WaEaETC302Q
CP
rTurnAGundam1isdefferentf
```

Output for the Sample Input

```
32Bad
AE86
7
EC302QTWaEa
TurnAGundam0isdefferentfr
```

Problem B
Lagrange's Four-Square Theorem
Input: lagrange.txt

The fact that any positive integer has a representation as the sum of at most four positive squares (*i.e.* squares of positive integers) is known as Lagrange's Four-Square Theorem. The first published proof of the theorem was given by Joseph-Louis Lagrange in 1770. Your mission however is not to explain the original proof nor to discover a new proof but to show that the theorem holds for some specific numbers by counting how many such possible representations there are.

For a given positive integer n, you should report the number of all representations of n as the sum of at most four positive squares. The order of addition does not matter, e.g. you should consider $4^2 + 3^2$ and $3^2 + 4^2$ are the same representation.

For example, let's check the case of 25. This integer has just three representations $1^2+2^2+2^2+4^2$, $3^2 + 4^2$, and 5^2. Thus you should report 3 in this case. Be careful not to count $4^2 + 3^2$ and $3^2 + 4^2$ separately.

Input

The input is composed of at most 255 lines, each containing a single positive integer less than 2^{15}, followed by a line containing a single zero. The last line is not a part of the input data.

Output

The output should be composed of lines, each containing a single integer. No other characters should appear in the output.

The output integer corresponding to the input integer n is the number of all representations of n as the sum of at most four positive squares.

Sample Input

```
1
25
2003
211
20007
0
```

Output for the Sample Input

```
1
3
48
7
738
```

Problem C
Area of Polygons
Input: area.txt

Yoko's math homework today was to calculate areas of polygons in the xy-plane. Vertices are all aligned to grid points (*i.e.* they have integer coordinates).

Your job is to help Yoko, not good either at math or at computer programming, get her homework done. A polygon is given by listing the coordinates of its vertices. Your program should approximate its area by counting the number of unit squares (whose vertices are also grid points) intersecting the polygon. Precisely, a unit square "intersects the polygon" if and only if the intersection of the two has non-zero area. In the figure below, dashed horizontal and vertical lines are grid lines, and solid lines are edges of the polygon. Shaded unit squares are considered intersecting the polygon. Your program should output 55 for this polygon (as you see, the number of shaded unit squares is 55).

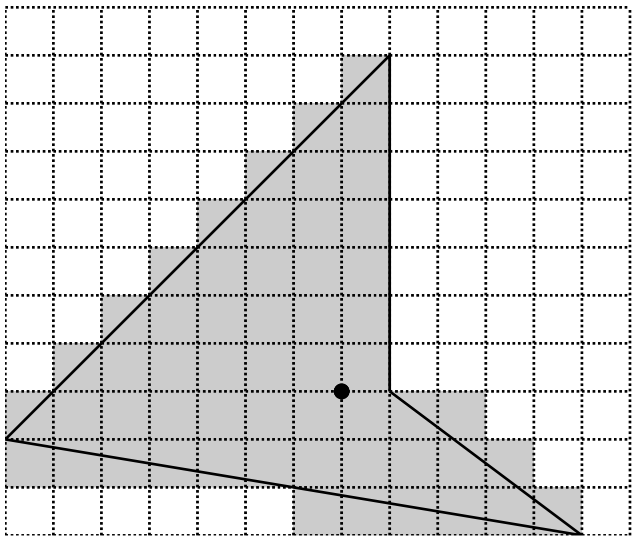


Figure 1: A polygon and unit squares intersecting it

Input

The input file describes polygons one after another, followed by a terminating line that only contains a single zero.

A description of a polygon begins with a line containing a single integer, m (≥ 3), that gives the number of its vertices. It is followed by m lines, each containing two integers x and y, the coordinates of a vertex. The x and y are separated by a single space. The i-th of these m lines gives the coordinates of the i-th vertex ($i = 1, \cdots, m$). For each $i = 1, \cdots, m-1$, the i-th vertex and the $(i+1)$-th vertex are connected by an edge. The m-th vertex and the first vertex are also connected by an edge (i.e., the curve is closed). Edges intersect only at vertices. No three edges share a single vertex (i.e., the curve is simple). The number of polygons is no more than 100. For each polygon, the number of vertices (m) is no more than 100. All coordinates x and y satisfy $-2000 \leq x \leq 2000$ and $-2000 \leq y \leq 2000$.

Output

The output should consist of as many lines as the number of polygons. The k-th output line should print an integer which is the area of the k-th polygon, approximated in the way described above. No other characters, including whitespaces, should be printed.

Sample Input

```
4
5 -3
1 0
1 7
-7 -1
3
5 5
18 5
5 10
3
-5 -5
-5 -10
-18 -10
5
0 0
20 2
11 1
21 2
2 0
0
```

Output for the Sample Input

```
55
41
41
23
```

Problem D
Weather Forecast
Input: weather.txt

You are the God of Wind.

By moving a big cloud around, you can decide the weather: it invariably rains under the cloud, and the sun shines everywhere else.

But you are a benign God: your goal is to give enough rain to every field in the countryside, and sun to markets and festivals. Small humans, in their poor vocabulary, only describe this as "weather forecast".

You are in charge of a small country, called Paccimc. This country is constituted of 4×4 square areas, denoted by their numbers.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Your cloud is of size 2×2, and may not cross the borders of the country.

You are given the schedule of markets and festivals in each area for a period of time.

On the first day of the period, it is raining in the central areas (6-7-10-11), independently of the schedule.

On each of the following days, you may move your cloud by 1 or 2 squares in one of the four cardinal directions (North, West, South, and East), or leave it in the same position. Diagonal moves are not allowed. All moves occur at the beginning of the day.

You should not leave an area without rain for a full week (that is, you are allowed at most 6 consecutive days without rain). You don't have to care about rain on days outside the period you were given: *i.e.* you can assume it rains on the whole country the day before the period, and the day after it finishes.

Input

The input is a sequence of data sets, followed by a terminating line containing only a zero.

A data set gives the number N of days (no more than 365) in the period on a single line, followed by N lines giving the schedule for markets and festivals. The i-th line gives the schedule for the i-th day. It is composed of 16 numbers, either 0 or 1, 0 standing for a normal day, and 1 a market or festival day. The numbers are separated by one or more spaces.

Output

The answer is a 0 or 1 on a single line for each data set, 1 if you can satisfy everybody, 0 if there is no way to do it.

Sample Input

```
1
0 0 0 0  0 1 0 0  0 0 0 0  0 0 0 0
7
0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0
1 0 0 0  0 0 1 0  0 0 0 1  1 0 0 1
0 0 0 0  0 0 0 0  1 0 0 0  0 1 0 1
0 0 0 0  0 0 0 0  0 1 0 1  0 0 0 0
0 1 0 1  0 0 0 0  0 0 0 0  0 0 0 0
1 0 0 1  0 0 0 0  0 0 0 0  0 0 0 1
0 0 0 0  0 1 0 0  1 0 0 0  0 0 0 0
7
0 0 0 0  0 0 0 0  1 0 0 0  0 0 0 0
0 0 1 0  0 0 0 1  0 0 0 0  0 1 0 0
0 0 0 1  0 0 0 0  0 0 1 0  1 0 0 0
0 1 0 0  0 0 0 1  0 0 0 0  1 0 0 0
0 0 0 0  0 0 0 0  1 0 0 0  0 0 0 0
0 0 0 0  0 0 0 1  1 0 1 0  0 0 0 1
0 0 0 0  0 0 0 0  0 0 0 1  0 0 0 0
15
0 0 0 0  0 0 0 0  0 0 0 0  0 0 1 0
0 0 0 0  0 0 0 0  1 0 0 0  0 0 0 0
0 0 0 0  0 0 0 0  1 1 0 0  0 0 0 0
0 0 0 0  0 0 0 0  1 0 0 0  0 0 0 0
0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0
0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0
0 0 0 0  0 0 0 0  1 1 0 1  0 0 0 0
0 0 0 0  0 0 0 0  0 0 0 0  1 0 0 0
0 0 1 1  0 0 0 0  0 1 0 0  0 0 0 0
1 1 0 0  0 0 0 0  0 0 1 0  0 1 0 0
0 0 0 0  0 1 0 0  0 0 0 1  0 0 0 0
0 0 1 0  0 0 0 0  0 0 0 0  0 0 1 0
1 0 0 1  1 0 0 0  0 1 0 1  0 0 0 0
0 0 0 0  0 0 0 0  0 0 0 0  0 0 1 0
0 0 0 0  0 1 0 1  0 1 0 0  0 0 0 0
0
```

Output for the Sample Input

```
0
1
0
1
```

Problem E
Molecular Formula
Input: mf.txt

Your mission in this problem is to write a computer program that manipulates molecular formulae in *virtual chemistry*. As in real chemistry, each molecular formula represents a molecule consisting of one or more atoms. However, it may not have chemical reality.

The following are the definitions of atomic symbols and molecular formulae you should consider.

- An atom in a molecule is represented by an atomic symbol, which is either a single capital letter or a capital letter followed by a small letter. For instance `H` and `He` are atomic symbols.
- A molecular formula is a non-empty sequence of atomic symbols. For instance, `HHHeHHHe` is a molecular formula, and represents a molecule consisting of four `H`'s and two `He`'s.
- For convenience, a repetition of the same sub-formula $\overbrace{X \ldots X}^{n}$, where n is an integer between 2 and 99 inclusive, can be abbreviated to `(`X`)`n. Parentheses can be omitted if X is an atomic symbol. For instance, `HHHeHHHe` is also written as `H2HeH2He`, `(HHHe)2`, `(H2He)2`, or even `((H)2He)2`.

The set of all molecular formulae can be viewed as a formal language. Summarizing the above description, the syntax of molecular formulae is defined as follows.

$$\begin{array}{rcl} \textit{Molecule} & \rightarrow & \textit{Atom} \mid \textit{Atom Number} \mid \texttt{(}\ \textit{Molecule}\ \texttt{)}\ \textit{Number} \mid \textit{Molecule Molecule} \\ \textit{Atom} & \rightarrow & \textit{CapitalLetter} \mid \textit{CapitalLetter SmallLetter} \\ \textit{Number} & \rightarrow & \texttt{2} \mid \texttt{3} \mid \texttt{4} \mid \cdots \mid \texttt{97} \mid \texttt{98} \mid \texttt{99} \\ \textit{CapitalLetter} & \rightarrow & \texttt{A} \mid \texttt{B} \mid \texttt{C} \mid \cdots \mid \texttt{X} \mid \texttt{Y} \mid \texttt{Z} \\ \textit{SmallLetter} & \rightarrow & \texttt{a} \mid \texttt{b} \mid \texttt{c} \mid \cdots \mid \texttt{x} \mid \texttt{y} \mid \texttt{z} \end{array}$$

Each atom in our virtual chemistry has its own atomic weight. Given the weights of atoms, your program should calculate the weight of a molecule represented by a molecular formula. The molecular weight is defined by the sum of the weights of the constituent atoms. For instance, assuming that the atomic weights of the atoms whose symbols are `H` and `He` are 1 and 4, respectively, the total weight of a molecule represented by `(H2He)2` is 12.

Input

The input consists of two parts. The first part, the Atomic Table, is composed of a number of lines, each line including an atomic symbol, one or more spaces, and its atomic weight which is a positive integer no more than 1000. No two lines include the same atomic symbol.

The first part ends with a line containing only the string `END_OF_FIRST_PART`.

The second part of the input is a sequence of lines. Each line is a molecular formula, not exceeding 80 characters, and contains no spaces. A molecule contains at most 10^5 atoms. Some atomic symbols in a molecular formula may not appear in the Atomic Table.

The sequence is followed by a line containing a single zero, indicating the end of the input.

Output

The output is a sequence of lines, one for each line of the second part of the input. Each line contains either an integer, the molecular weight for a given molecular formula in the corresponding input line if all its atomic symbols appear in the Atomic Table, or `UNKNOWN` otherwise. No extra characters are allowed.

Sample Input

```
H  1
He 4
C  12
O  16
F  19
Ne 20
Cu 64
Cc 333
END_OF_FIRST_PART
H2C
(MgF)2As
Cu(OH)2
H((CO)2F)99
0
```

Output for the Sample Input

```
14
UNKNOWN
98
7426
```

Problem F
Gap
Input: gap.txt

Let's play a card game called Gap.

You have 28 cards labeled with two-digit numbers. The first digit (from 1 to 4) represents the suit of the card, and the second digit (from 1 to 7) represents the value of the card.

First, you shuffle the cards and lay them face up on the table in four rows of seven cards, leaving a space of one card at the extreme left of each row. The following shows an example of initial layout.

	42	21	13	22	32	26	23
	16	43	47	14	24	34	36
	46	17	27	31	11	37	12
	35	41	44	25	15	33	45

Next, you remove all cards of value 1, and put them in the open space at the left end of the rows: "11" to the top row, "21" to the next, and so on.

Now you have 28 cards and four spaces, called gaps, in four rows and eight columns. You start moving cards from this layout.

11	42		13	22	32	26	23
21	16	43	47	14	24	34	36
31	46	17	27			37	12
41	35		44	25	15	33	45

At each move, you choose one of the four gaps and fill it with the successor of the left neighbor of the gap. The successor of a card is the next card in the same suit, when it exists. For instance the successor of "42" is "43", and "27" has no successor.

In the above layout, you can move "43" to the gap at the right of "42", or "36" to the gap at the right of "35". If you move "43", a new gap is generated to the right of "16". You cannot move any card to the right of a card of value 7, nor to the right of a gap.

The goal of the game is, by choosing clever moves, to make four ascending sequences of the same suit, as follows.

11	12	13	14	15	16	17	
21	22	23	24	25	26	27	
31	32	33	34	35	36	37	
41	42	43	44	45	46	47	

Your task is to find the minimum number of moves to reach the goal layout.

Input

The input starts with a line containing the number of initial layouts that follow.

Each layout consists of five lines — a blank line and four lines which represent initial layouts of four rows. Each row has seven two-digit numbers which correspond to the cards.

Output

For each initial layout, produce a line with the minimum number of moves to reach the goal layout. Note that this number should not include the initial four moves of the cards of value 1. If there is no move sequence from the initial layout to the goal layout, produce "-1".

Sample Input

```
4

12 13 14 15 16 17 21
22 23 24 25 26 27 31
32 33 34 35 36 37 41
42 43 44 45 46 47 11

26 31 13 44 21 24 42
17 45 23 25 41 36 11
46 34 14 12 37 32 47
16 43 27 35 22 33 15

17 12 16 13 15 14 11
27 22 26 23 25 24 21
37 32 36 33 35 34 31
47 42 46 43 45 44 41

27 14 22 35 32 46 33
13 17 36 24 44 21 15
43 16 45 47 23 11 26
25 37 41 34 42 12 31
```

Output for the Sample Input

```
0
33
60
-1
```

Problem G
Concert Hall Scheduling
Input: hall.txt

You are appointed director of a famous concert hall, to save it from bankruptcy. The hall is very popular, and receives many requests to use its two fine rooms, but unfortunately the previous director was not very efficient, and it has been losing money for many years. The two rooms are of the same size and arrangement. Therefore, each applicant wishing to hold a concert asks for a room without specifying which. Each room can be used for only one concert per day.

In order to make more money, you have decided to abandon the previous fixed price policy, and rather let applicants specify the price they are ready to pay. Each application shall specify a period $[i, j]$ and an asking price w, where i and j are respectively the first and last days of the period ($1 \leq i \leq j \leq 365$), and w is a positive integer in yen, indicating the amount the applicant is willing to pay to use a room for the whole period.

You have received applications for the next year, and you should now choose the applications you will accept. Each application should be either accepted for its whole period or completely rejected. Each concert should use the same room during the whole applied period.

Considering the dire economic situation of the concert hall, artistic quality is to be ignored, and you should just try to maximize the total income for the whole year by accepting the most profitable applications.

Input

The input has multiple data sets, each starting with a line consisting of a single integer n, the number of applications in the data set. Then, it is followed by n lines, each of which represents one application with a period $[i, j]$ and an asking price w yen in the following format.

$$i \quad j \quad w$$

A line containing a single zero indicates the end of the input.

The maximum number of applications in a data set is one thousand, and the maximum asking price is one million yen.

Output

For each data set, print a single line containing an integer, the maximum total income in yen for the data set.

Sample Input

```
4
1 2 10
2 3 10
3 3 10
1 3 10
6
1 20 1000
3 25 10000
5 15 5000
22 300 5500
10 295 9000
7 7 6000
8
32 251 2261
123 281 1339
211 235 5641
162 217 7273
22 139 7851
194 198 9190
119 274 878
122 173 8640
0
```

Output for the Sample Input

```
30
25500
38595
```

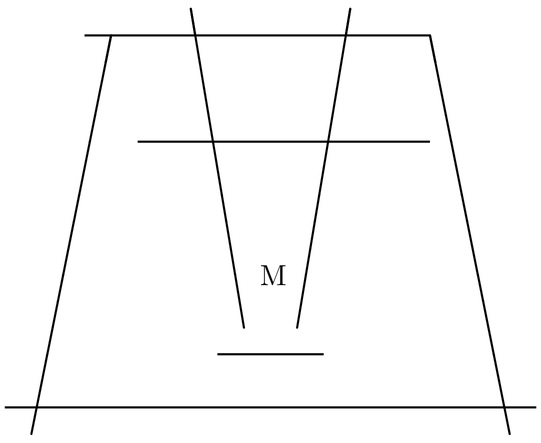
Problem H
Monster Trap
Input: trap.txt

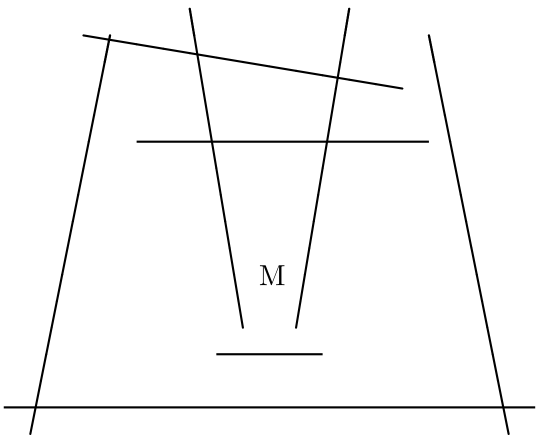
Once upon a time when people still believed in magic, there was a great wizard Aranyaka Gondlir. After twenty years of hard training in a deep forest, he had finally mastered ultimate magic, and decided to leave the forest for his home.

Arriving at his home village, Aranyaka was very surprised at the extraordinary desolation. A gloom had settled over the village. Even the whisper of the wind could scare villagers. It was a mere shadow of what it had been.

What had happened? Soon he recognized a sure sign of an evil monster that is immortal. Even the great wizard could not kill it, and so he resolved to seal it with magic. Aranyaka could cast a spell to create a monster trap: once he had drawn a line on the ground with his magic rod, the line would function as a barrier wall that any monster could not get over. Since he could only draw straight lines, he had to draw several lines to complete a monster trap, i.e., magic barrier walls enclosing the monster. If there was a gap between barrier walls, the monster could easily run away through the gap.

For instance, a complete monster trap without any gaps is built by the barrier walls in the left figure, where “M” indicates the position of the monster. In contrast, the barrier walls in the right figure have a loophole, even though it is almost complete.





Your mission is to write a program to tell whether or not the wizard has successfully sealed the monster.

Input

The input consists of multiple data sets, each in the following format.

$$\begin{array}{llll} n & & & \\ x_1 & y_1 & x'_1 & y'_1 \\ x_2 & y_2 & x'_2 & y'_2 \\ & & \ldots & \\ x_n & y_n & x'_n & y'_n \end{array}$$

The first line of a data set contains a positive integer n, which is the number of the line segments drawn by the wizard. Each of the following n input lines contains four integers x, y, x', and y', which represent the x- and y-coordinates of two points (x, y) and (x', y') connected by a line segment. You may assume that all line segments have non-zero lengths. You may also assume that n is less than or equal to 100 and that all coordinates are between -50 and 50, inclusive.

For your convenience, the coordinate system is arranged so that the monster is always on the origin $(0, 0)$. The wizard never draws lines crossing $(0, 0)$.

You may assume that any two line segments have at most one intersection point and that no three line segments share the same intersection point. You may also assume that the distance between any two intersection points is greater than 10^{-5}.

An input line containing a zero indicates the end of the input.

Output

For each data set, print "`yes`" or "`no`" in a line. If a monster trap is completed, print "`yes`". Otherwise, i.e., if there is a loophole, print "`no`".

Sample Input

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

```
6 9 9 -6
-1 -2 -3 10
1 -2 3 10
-2 -3 2 -3
0
```

Output for the Sample Input

```
yes
no
```