ACM PROGRAMMING CONTEST
SOUTH PACIFIC REGION
1998

GUI Problem **ProblemSolving**

Note: this problem will not count as a "solved problem" for thepurposes of your total score. Instead, it will be scored as follows:
• You can make as many submissions as you want until youget it correct.
• At that point, you will be told that it is correct and you mustmake no more submissions.
• The judges will assess your program's user interface,either at the time of accepting it or some time later, and will assign ascore from 0 to 10 depending on its quality.
• Your final score for the program will be -(30 + 3*(quality score))minutes; that is, between 30 and 60 minutes will be taken off the totaltime for your other problems.

Suppose you work for a firm which has a lot of customers, andoccasionally some of these customers have problems. Your firm has decidedto deal with these problems in a standardised way: in fact, they havedetermined thatthe customer must be told one of the following five responses:

• The firm will investigate the problem further.
• He or she is imagining things - there is nothing wrong, as far asthe firm can see.
• It is the fault of the electricity supplier, Mercury Energy.
• The customer is not in the database so the firm has no record ofany transaction.
• The firm will give the customer their money back.

Your previous attempt at computerising this approach, anartificial intelligence program, has been a complete failure as it alwaysgave the customer their money back. The current plan is to have anoperator sit in front of a screen and be presented with a description of the problem. The operator mustthen decide which of the five responses to give. For example, with atext-based system, they could type "D" for "investigate further", "F" for "nothing wrong", "B" for "Mercury Energy", etc. Your system, however, mustuse a well-designed Graphical User Interface that anyone can use with notraining.

Input will be from a file called GUI.dat and will cosist of anumber of customer problems. Each customer problem will be given by one ormore sentences of text - from 20 to 500 characters - terminated by astandard end-of-line sequence. Thetext for each problem must be displayed on the screen in such a way thatthe operator can read it (several times if necessary), and then take someaction which will determine the solution (eg select something on the screen). For each of the five possible solutions it must be immediately obvious which action to take. When an action is taken, the problem must be cleared offthe screen and the next one displayed. The end of the placement problemswill be indicated by aproblem statement consisting of a single #. Your program must terminatewhen it reads this end statement.

There is no output. Only the user interface is required, so theresponse is "thrown away" after being entered.

Problem A **Findingwords**

A common problem when processing incoming text is to isolate thewords in the text. This is made more difficult by the punctuation; wordshave commas, "quote marks", (even brackets) next to them, or hyphens in the middle of theword. This punctuation doesn't count as letters when the words have to belooked up in a dictionary by the program.

For this problem, you must separate out "clean" words from text, that is,words with no attached or embedded non-letters. A "word" is any continuousstring of non-whitespace characters, with whitespace characters on eachside of it. For this problem, a "whitespace" character is a space character or an end-of-line character, orthe start or end of the file (so that, for example, if the file is "A B",where there is a space character between the A and B but no other, thenthere are two words, "A" and "B").

Input will be from standard input (ie the keyboard or a redirected file)and will consist of lines with no more than 60 characters in each line.Every line will be terminated by a character which isn't whitespace (whichwill be followed immediately by anend-of-line character). The input will be terminated by a line consistingof a single #.

Output, which must be written to standard output (the screen), must be thelines of the incoming text, with the punctuation stripped away from eachword. "Punctuation" is any character which is not a letter (a - z and A -Z) or a whitespace character - your program must not change the letters and space characters (although spacecharacters at the ends of lines will be ignored by the judges). Whenpunctuation occurs in the middle of a word (ie there is no whitespacecharacternext to it), it must be simply removed - see what happens to the word"doesn't" in the example. A word which consists entirely of punctuationwill therefore be removed entirely. There is a special rule for a hyphen("-") which is the very last characterin a line - the word part before the hyphen, and the first word part on thenext line, form a single word, and this must be written on a line by itself(so that no line is ever lengthened). There will always be a space beforethe word part on the first line, and a space after the word part on the second line.

EXAMPLE

Input

```
A common problem when processing incoming text is to isolate
the words in the text.  This is made more difficult by the
punctuation; words have commas, "quote marks",
(even brackets)      next to them, or hy-
phens in the middle of the word.  This punctuation doesn't
count as letters when the words have to be looked up in a
# dictionary by the 12345 "**&! program.
#
```

Output

```
A common problem when processing incoming text is to isolate
the words in the text  This is made more difficult by the
punctuation words have commas quote marks
even brackets      next to them or
hyphens
 in the middle of the word  This punctuation doesnt
count as letters when the words have to be looked up in a
 dictionary by the   program
```

Problem B **Y3KProblem**

We have heard a lot recently about the Y2K problem. According tothe doom sayers, planes will fall out of the skies, businesses will crashand the world will enter a major depression as the bugs in software andhardware bite hard. While this panic is a very satisfactory state of affairs for thecomputing profession, since it is leading to lots of lucrative jobs, itwill have a tendency to bring the profession into disrepute when almost noproblems occur on 1/1/00. To helpavoid this unseemly behaviour on any future occasion, you must write aprogram which will give the correct date for (almost) any number offuture days - in particular, it must correctly predict the date N daysahead of any given date, where N is a number less than 1,000,000,000 andthe given date is any date before the year 3000.

Remember that, in the standard Gregorian calendar we use, there are 365 days in a year except for leap years, when there are 366.Leap years are all years divisible by 4 and not 100, except that thosedivisible by 400 are leap years - thus 1900 was not a leap year, 1904, 1908... 1996 were leap years, 2000 will be a leap year, 2100 will not be a leap year, etc. The number of days ineach month in a normal year is 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,31; in a leap year, the second month has 29 days.

Input will be from standard input (ie the keyboard or aredirected file) and will consist of lines containing four numbers,separated by one or more spaces. The first number on each line will be thenumber of days you have to predict (between0 and 999999999), followed by the date in the format DD MM YYYY where DD isthe day of the month (1 to 31), MM is the month (1 to 12), and YYYY is theyear (1998 to 2999). The input will be terminated by a line containingfour 0's.

Output, which must be written to standard output (the screen),must be one line for each line of input, giving the date which is therequired number of days ahead of the date in the input line, in the sameformat as the dates for the input.

EXAMPLE

Input

```
1 31 12 2999
40 1 2 2004
60 31 12 1999
60 31 12 2999
146097 31 12 1999
999999 1 1 2000
0 0 0 0
```

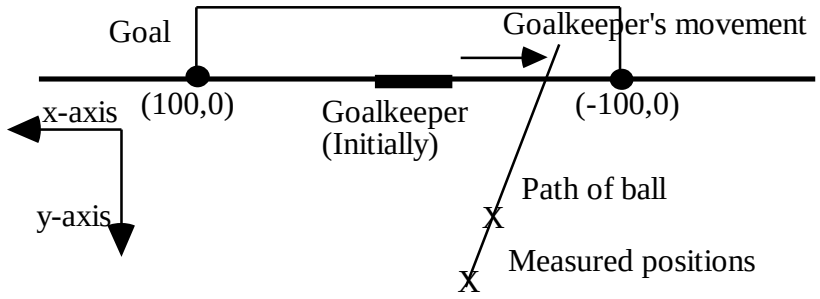
Output

```
1 1 3000
12 3 2004
29 2 2000
1 3 3000
31 12 2399
27 11 4737
0
```

Problem C **World CupSuccess**

A certain country, dissatisfied with their team's performance inthe Soccer World Cup, has decided to ensure success next time by automatingtheir goalkeeper's performance. The plan is to have a computer track the position of the ball at all times, and if the trajectory of the ball issuch that it will enter the goal mouth, then the movement the goalkeeperneeds to make will be computed, and a signal sent to the goalkeeper, who will be secretly wired so that theincoming signal will trigger a voltage pulse to his leg muscles and thespasm will cause him to move in the right direction with the rightacceleration. Your job is to compute the necessary movement to block the goal, in termsof direction and acceleration.

For this initial trial, only balls at ground level will beconsidered, and the goalkeeper's movement will be considered to be auniform acceleration from a standing position. The goalkeeper will berepresented by a line, one endof which must be at the exact spot the ball would cross the goal mouth, atthe exact time this would happen. The trajectory of the ball is found bymeasuring its position at two points - the straight line through these points then defines the path of theball. For the purposes of computation, a coordinate system is used asshown, with the (0,0) point being the centre of the goalkeeper's initialposition. The goalkeeper is 40 units wide.



Input will be from standard input (ie the keyboard or aredirected file) and will consist of lines each describing an attempt atgoal. There will be six integers on the line; the first two give the (x,y) coordinates of the first point on the path of the ball, thesecond two give the (x,y) coordinates of the second point on the path ofthe ball, the fifth number gives the number of millisecond between thefirst point observation and the second point observation, and the sixth number gives the number ofmilliseconds from the first point observation until the goalkeeper startsto move. The fifth and sixth numbers will always be positive with thesixth number greater than the 5th number. The path of the ball will always be such that the ball will enter thegoal unless stopped by the goalkeeper (ie the y-coordinate of the secondpoint will always be less than the y-coordinate of the first point, and thex-coordinate of thepoint where the line crosses the x-axis will be between -100 and 100). Theinput will be terminated by a line consisting of six 0's.

You must work out the constant acceleration the goalkeeper mustmake in order for his closest edge to be at the exact point the ball's pathcrosses the x-axis, at the exact time the ball reaches that point.The acceleration must be positive if the goalkeeper must move to his right,and negative if he must move to his left (actually this isn't correctmathematically - the acceleration will always be positive - but it is usedto indicate direction).Note that if the ball's path intersects the original position of thegoalkeeper then the acceleration must be 0. The following three formulae(for uniformly accelerated objects which are initially stationary) may beof assistance to those who have forgotten their elementary Physics:

speed = acceleration x time

distance = 0.5 x acceleration x time^2

speed^2 = 2 x acceleration x distance

Output, which must be written to standard output (the screen),must be one line for each line of input, giving the constant accelerationneeded to two decimal places. There will always be a finite answer, that is, the time the goalkeeper starts to move will be at least 1ms before the ball arrives at the x-axis.

EXAMPLE

Input	**Output**
-10 100 -20 80 2 4	-2.22
0 150 6 100 1 2	0.00
0 0 0 0 0 0	

Problem D **WordProblem**

In a popular puzzle, often found in newspapers, a set of letters isprovided, and the challenge is to find how many different words can be madefrom these letters. This problem is designed to take all the fun out of itby automating the process.

Input will be from standard input (ie the keyboard or aredirected file) and will be in two parts. The first part will be thedictionary which is to be used for the problem. It will have less than1000 lines, and each line will contain one word of up to 10 characters in lowercase.The words will be in alphabetic order. The end of the dictionary will beindicated by a line consisting of a single # character. After thedictionary there will bedata for several puzzles, each on a separate line. Each puzzle data linewill have from one to 7 lowercase letters, separated by one or more spaces.The list of puzzles will be terminated by a line consisting of a single#.

Output, which must be written to standard output (the screen),must be one line for each puzzle data line in the input, giving the numberof different words in the dictionary which can be formed by arranging someletters selected from the letters in the puzzle line.

EXAMPLE

Input

```
ant
bee
cat
dog
ewe
fly
gnu
#
b e w
b b e e w w
t a n c u g d
#
```

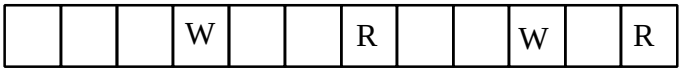
Output

```
0
2
3
```

Problem E **BoardGame**

A surprisingly complex game can be played between two players ona simple one-dimensional board with up to 60 holes; each player hascounters of one colour (indicatedhere by letters like W or R) which go into the holes. Players can move anyone counter any number of holes, up to the next counter on the board (counters may not bejumped, only empty holes). The object of the game is to block the otherplayer so he or she has no allowable moves. With one counter each, thefirst player can force a win on the first move by moving their counter next to the other counter.Whenever the second player tries to move away, the first player moves next to them. Eventually the second player will have no moves left. Ifthe first player does not take this first move, then the second player canforce a win. Convince yourself that these statements are true in thisexample:

With two counters each, the game is more complex. For example,in the following situation, if W moves first they can force a win by movingthe leftmost counter one square to the right, or the rightmost counter onesquare to the left. Any other move (for example, moving one of the W counters next to an Rcounter) will mean than R can force a win. Try it and convince yourselfthis is true.



Even more complex situations arise with more counters; forexample, draws can occur (ie neither player can force a win) as in thefollowing case:

It is embarrassing that it is difficult to see how to win such asimple game. Please write a program so that I can play this safely byalways knowing the winning move, or, when there is no winning move, whetherthe game is lost or can bedrawn. Assume each player always makes the best move.

Input will be from standard input (ie the keyboard or aredirected file) and will consist of lines each containing a gamedescription, which will be a string of up to 60 digits. Each digit will be 0, for an empty hole, or1, for a counter belonging to the first player to move, or 2, for a counterbelonging to the second player to move. Both players will have the samenumber of counters, and the total number will always be at least two less than the totalnumber of holes in the board. The input will be terminated by a lineconsisting of a single zero.

Output, which must be written to standard output (the screen),must be one line for each line of input. This line must either be 0, ifthe game cannot be won but can be drawn, or 2, if the first player willlose whatever move is made, or a 1, followed by a description of the winning move. The move description mustbe given by two numbers, the first giving the hole number the piece to movestarts in, and the second giving the hole number it is to be moved to,where the holes are assumed to be numbered from left to right with the leftmost being number 1. If thereis more than one winning move, give the one which involves a move from thesmallest numbered hole; if there is still more than one move possible, give the one which involves a move to the smallest numbered hole.

EXAMPLE

Input

```
000200100000
000100200102
000010201002
000010200102
001020020001100002000
0
```

Output

```
1 7 5
1 4 5
1 5 4
2
1 13 15
```

Problem F **Anagrammatic Primes**

A number is "prime" if it has no divisors other than itself and1. For example, 23 is prime and 35 is not prime because 35 = 7 x 5. Ifthe digits of a number are rearranged, then usually its primeness changes - for example, 32 is not prime but 53 is. For thisproblem, you have to find numbers which are prime no matter how yourearrange their digits. For example, all of the numbers 113, 131 and 311are prime, so we say that 113 is an "anagrammatic prime" (also 131 and 311 are anagrammatic primes).

Input will be from standard input (ie the keyboard or aredirected file) and will consist of lines with a single positive number,n, less than 10,000,000, on each line. You must find the firstanagrammatic prime which is bigger than n and less than the next power of 10 above n. The input will be terminatedby a line consisting of a single 0.

Output, which must be written to standard output (the screen),must be one number for each number in the input. The number must either bethe next anagrammatic prime, as described above, or 0 if there is noanagrammatic prime in the range defined.

EXAMPLE

Input

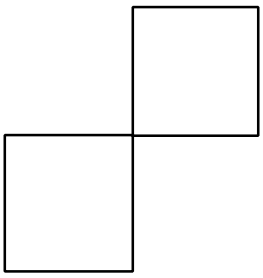
```
10
16
900
113
8000000
0
```

Output

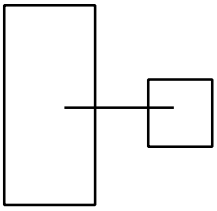
```
11
17
919
131
0
```

Problem G **Hole Cutter** 150 points

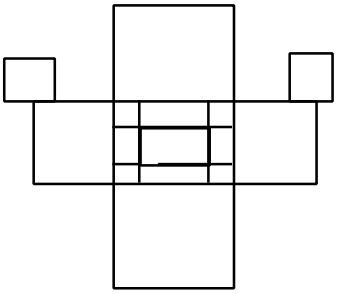
A factory which specialises in making cuts in the interior offlat sheets has just acquired a new cutter which can make cuts much morefreely than any of their previous machines, and they want you to write aprogram tocalculate exactly what has happened when a complex series of cuts are made.In particular, they need to know the number of holes which are formed inthe sheet by the cuts. Here are some examples of situations that can ariseafter cutting:



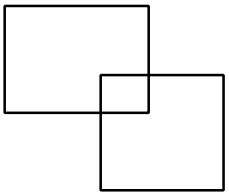
Two holes



Two holes



One hole



One hole

Input will be from standard input (ie the keyboard or aredirected file) and will consist of several cutting operationdescriptions. Each description starts with a number, N, giving the numberof cuts in the operation, followed by N lines giving the actual cuts. The number N will always beless than 100 and greater than 1. Each cut will be given by four wholenumbers separated by one or more spaces, the first two giving the (x,y)coordinates of the start of the cut line and the second two defining the end of the cut line; thecoordinate values will always be whole numbers less than 10000. You shouldassume the points are always internal to the sheet, never on the boundary.Each cut willbe parallel to the x or y axis of the table. The input will be terminatedby a line consisting of a single 0, ie a cutting operation description withN = 0. The first example given below describes the lefthand pictureabove.

Output, which must be written to standard output (the screen),must be one number for each cutting operation description in the input,giving the number of distinct holes in the sheet after the cuts. Note thatthe minimum area of any hole is 1square unit.

EXAMPLE

Input

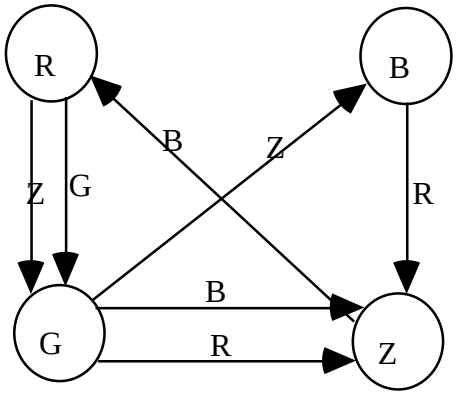
```
6
1 0 1 1
2 0 2 2
3 1 3 2
1 0 2 0
1 1 3 1
2 2 3 2
2
0 1 2 1
1 2 1 0
0
```

Output

```
2
0
```

Problem H Colour Circles

A very colourful one-person game can be played as follows.First a set of colours is selected then a set of circles is drawn usingsome or all of the colours, with duplicates possible - there are at leastas many circles as colours. These circles are then connected together in any way by coloured arrows- any number of arrows, with any colours, may be used to connect any pairof circles. For example, if we use the four colours R, G, B, and Z andfour circles then we could have the following situation:



Three different circles are then picked from the set; two ofthem have a counter placed inside, while the third is the "target" circle.A counter may be moved from one circle to another along an arrow (in thecorrect direction), only if the other counter is not in the circle being moved to, and the colour of thearrow is the same as the colour of the circle the other counter is in. Acounter may be moved several times in succession - they don't have to bemoved alternately. The aim is to get one of the counters in the target circle, in the least number ofmoves; if the target circle can't be reached, the game is"impossible".

For example, in the picture above, if one counter is in theB circle, the other counter is in the Z circle, and the target is the Gcircle, then the game can only be won by moving the Z counter to the Rcircle (since a B arrow runs in that direction), which makes it possible to move the B counter to the Zcircle along the R arrow, and the R counter can now be moved to the Gcircle along the Z arrow, for a total of three moves.

Input will be from standard input (ie the keyboard or a redirected file)and will consist of descriptions of several games, using numbers instead ofcolours. The first line of each game description contains five numbers, N,R, S, T, M where N is the number of circles in the game (they will be numbered 1 toN, with N <= 100), R and S are the numbers of the circles the two countersstart in, T is the number of the target circle, and M is the total numberof arrowsconnecting the circles (M <= 5,000). After this are several lines (maximumlength 60 characters) containing N numbers giving the colours of thecircles in order (1 to N), with up to 20 numbers per line, separated by oneor more spaces. The colours are denoted by numbers from 1 to N - some of these numbersmay be unused. Then come M lines which define the arrows, in no particularorder. Each contains three numbers; the first is the number of thestarting circle, the second the number of the ending circle, and the third is the colour of the line. The inputwill be terminated by a line consisting of a five zeroes. The firstexample below describes the picture above.

Output, which must be written to standard output (thescreen), must be one number for each game description giving the minimumnumber of moves to complete the game, or 0 if the game is impossible.

EXAMPLE

Input

```
4 2 3 4 7
1 2 3 4
1 4 3
1 4 4
2 3 1
3 1 2
```

Input Continued

```
4 1 5
4 5 3
5 1 4
3 2 1
3 2 2
5 3 3
```

```
4 2 3
4 3 2
4 3 1
5 3 4 1 8
2 3 2 1 4
2 1 2
```

```
3 5 1
0 0 0 0 0
```

Output

```
3
4
```