1989 ACM Scholastic Programming Contest
Problem "A"
Finger Walks

Your employer, a popular telecommunications company, is thinking about offering a special service to avid chess players wherein they could all have their telephone numbers in the same ubiquitous area code. Since telephone numbers cannot begin with 0 or 1, that area code could encompass 8,000,000 different seven-digit telephone numbers. That's far more than needed for this customer group, so the Marketing Department is considering the use of extra added attractions in the assignments of numbers to customers. They have requested a program which determines the counts of usable telephone numbers having certain properties.

For the keypad layout illustrated, the program is to calculate and print (labeled) the counts of allowable seven-digit telephone numbers in which:

a) Consecutive digits are a knight's move apart on the keypad; (6 & 8 can follow 1; 7 & 9 can follow 2; 4 & 8 can follow 3; 3, 9, & 0 can follow 4; etc.)

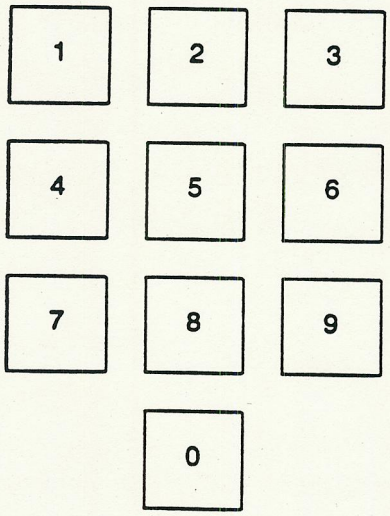
b) Consecutive digits are a bishop's move apart on the keypad; (Adjacent pairs of digits are unequal but lie on the same diagonal.)

c) Consecutive digits are a rook's move apart on the keypad; (Adjacent pairs of digits are unequal but lie on the same row or column.)

d) Consecutive digits are a king's move apart on the keypad; (Adjacent pairs of digits are unequal but are horizontally, vertically, or diagonally adjacent on the keypad.)

e) Consecutive digits are a queen's move apart on the keypad; (Adjacent pairs of digits are unequal but lie on the same row, column, or diagonal.)

Because there is not yet any revenue from this service to charge computer time against, the program cannot run long enough to sieve all 8,000,000 numbers. Nor can it run long enough to recursively traverse seven-level trees to count leaf nodes.



KEYPAD LAYOUT

1989 ACM Scholastic Programming Contest
Problem "B"
Communicating Boats

You are the owner of a number of fishing boats operating in and around the waters of an island in the South Pacific. You desire to communicate from your command boat to a forward "scouting" boat via the shortest path (i.e., the fewest number of intermediate boats).

The problem is that any two boats in the fleet can communicate directly with one another if and only if they are in line of sight of each other. That is, no part of the island may come between directly-communicating boats. If boats are not in line of sight of each other then they must communicate by routine their messages through intermediate boats. Your program should compute the "shortest" path between the command boat and the scout boat; that is, the path requiring the least number of intermediate boats. Note that this may not be the path of shortest linear distance. Once computed, your program should output this path. For each data set, it may *always* be assumed that at least one path exists.

Input

Read in the number of boats (an integer value in the range 2 through 10) together with the radius of an island (the radius is a real value greater than 0.0). Next read in an ordered pair of real (x,y) coordinates for each boat in the fleet. Every boat will be "outside" the island, which is centered at the origin. The first pair of coordinates are those of the command boat, while the last pair are those of the scouting boat. Thus, for a given data set, the first boat is always trying to communicate with the last. For this problem there are multiple data sets. The end-of-data will be signaled when a value of 0 boats and a radius of 0.0 is encountered as input.

Output

For each data set read, your program should display the shortest path to communicate from the command boat to the scout boat.

1989 ACM Scholastic Programming Contest
Problem "C"
The Thief of Bagdad

The Bagdad Office Building is on fire. On each floor of this high-rise building (at most 150 floors) there is a sack of gold coins. Your task is to determine the maximum number of coins that can be safely retrieved subject to the following conditions:

1. You must use an elevator which moves at a speed of 10 floors per minute.
2. You are in the elevator on the first floor when the fire starts.
3. The elevator cannot safely pass through a floor which is on fire.
4. It takes you 10 seconds to retrieve the coins on any floor.
5. The fire spreads down at a rate of 1 floor per minute.

You are not able to safely retrieve the coins if the fire will arrive at that floor while you are there. Once you have finished retrieving the coins on any floor, you are assumed to have left that floor.

Input

Input consists of an integer specifying the floor on which the fire starts followed by pairs of integers specifying a floor number and the number of coins on that floor. Other floors are assumed to have no coins. A pair of zeroes indicates the end of input.

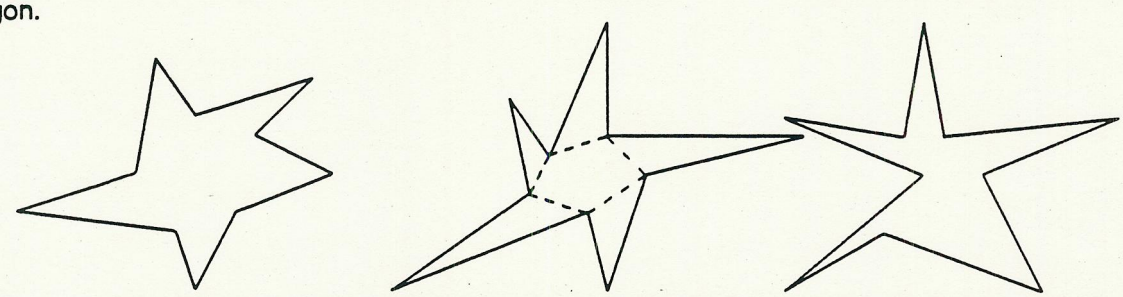
Output

Your program should display the number of coins retrieved.

1989 ACM Scholastic Programming Contest
Problem "D"
Star Painters

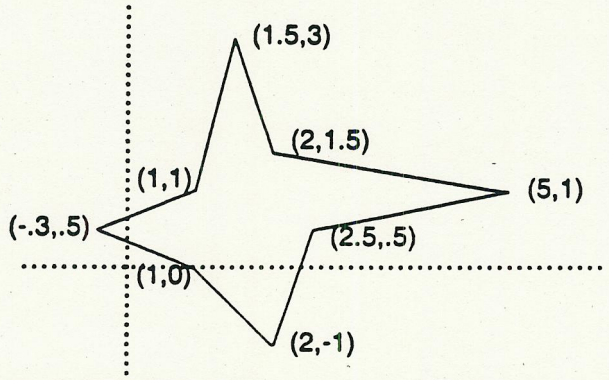
The ABC Theater Company prides itself in having spectacular stage props, but worries about its limited budget. Its next production requires a backdrop of gold stars suspended on a curtain. They already own the curtain, but they must cut the stars from cardboard (cheap) and paint them with gold gilt (expensive). They need to know the areas of the stars to determine how much gilt to purchase and whether to eliminate some starts to decrease expenses. You must write a program to calculate the area of each star.

Each star consists of a polygonal body (the "inside") with trangular rays coming out from each side of the body. Each body is convex (any two points inside the body can be connected by a line segment that lies inside the body). Each ray is a triangle, one side of which is a side of the body. No two rays of a star overlap. The examples below illustrate two typical stars, one with its body in outline, and a "star" which is unacceptable because its body is not a convex polygon.



A Typical Star Typical Star Showing Outlined Body An Unacceptable Star

A star is represented by a sequence of pairs of numbers which is a consecutive listing of its vertices in the real plane. Each sequence begins and ends with the same vertex, which is also a vertex on the star's body. That particular vertex is the only one that is repeated in the star's sequence. The input file consists of one or more star sequences and terminates with end-of-file. For example, the input `1 1 1.5 3 2 1.5 5 1 2.5 .5 2 -1 1 0 -.3 .5 1 1` represents the following star:



For each star, your program must output a line containing a header identifying the star's number in the input stream (star #1, star #2, etc.) and the star's area, correct to three decimal places. For example, the input file above should generate the following output:

```
Area of star #1 is 5.275
```

1989 ACM Scholastic Programming Contest
Problem "E"
Network Routing

Every Internet host machine holds a routing table to help determine where to send messages destined for other machines. A simplified model of this situation is a table which holds the data in pairs (Network number, Gateway host machine to route to for this network). Every network is identified by a 4 octet number (32 bits) broken into 4 fields. Using this scheme, different classes of networks are assigned unique numbers.

The network number for a class A network uses the first field for the network, and contains 0's in the remaining 3 fields. A class B network number uses the first 2 fields to identify the network and contains 0's in the last 2 fields. A class C network number uses the first 3 fields to identify the network, with 0 in the final field.

A host in a network is identified by its network number and its machine number. The machine number uses the octets not required by the network number. The scheme for determining network numbers and host machine numbers is shown in the following table. The letter N represents an additional component of the network number, and the letter H represents a component of the host machine number.

Class	Network Number	Host Number
A	(1-127).0.0.0	(1-127).H.H.H
B	(128-191).N.0.0	(128-191).N.H.H
C	(192-223).N.N.0	(192-223).N.N.H

In this problem you will build a simple version of a routing program which accepts only four commands. The details of the input and expected responses appear below. Your program is to process add requests (add a network/gateway pair to the table), delete requests (delete a network/gatway pair from the table), and route information requests (determine which gateway to send information for a particular host). Given a routing request, you will have to transform the host number into the appropriate network number in order to search the routing table.

Your program should process the input one command at a time and display its results as it executes, indicating sucess or failure for each of the add/delete inputs, as well as the proper routing or error indication for each routing request.

Input Data

The data file your program will process consists of four types of command lines, with data on each starting in column 1. Each successive part of the line is separated by one or more blanks. Trailing blanks on each line are to be ignored. Network and host numbers will octets. The input data will be valid (that is, no invalid commands, no funny octet values, and no poorly constructed host or network numbers).

CONTINUED ON REVERSE

Command Lines and Required Processing

`ADD NetworkNumber GatewayHostNumber`

Add a route - Lines of this form in the data file will give you a route to add to the table. Report an error if the network number given already has an entry in the table; otherwise, report success.

`DEL NetworkNumber GatewayHostNumber`

Delete a route - Lines of this form in the data file will give you a route to delete from the table. Report an error if the network and gateway host pair are not in the current table; otherwise, report success.

`RTE HostNumber`

Lookup a route for a host - Lines of this form in the data file require finding the proper gateway for the given host number. Convert the host number to the appropriate network number for the table lookup. Based on its network class, replace the appropriate octet(s) with zero(es). Report an error if the network for this host is not in the table. Otherwise, report success along with the gateway machine to which the information will be routed.

`END`

End program - A single line of this form will appear at the end of the data file.

Example Input

```
ADD     192.5.25.0  129.65.6.1
ADD     128.125.0.0  130.201.5.3
ADD     10.0.0.0   10.1.4.3
ADD     192.5.25.0  128.17.3.4
DEL     192.8.1.0 127.15.32.2
RTE     192.5.25.14
RTE     128.125.4.3
RTE     128.124.1.6
RTE     128.125.8.4
DEL     10.0.0.0  10.1.4.6
RTE     10.25.1.4
DEL     10.0.0.0   10.1.4.3
RTE     10.25.1.4
END
```

Example Output

```
Added route: 192.5.25.0 - 129.65.6.1
Added route: 128.125.0.0 - 130.201.5.3
Added route: 10.0.0.0 - 10.1.4.3
Add error: 195.5.25.0 already exists in table
Delete error: 192.8.1.0 - 127.15.32.2 doesn't exist in table
Route 192.5.25.14 through 129.65.6.1
Route 128.125.4.3 through 130.201.5.3
Route 128.124.1.6 cannot be determined from table
Route 128.125.8.4 through 130.201.5.3
Delete error: 10.0.0.0 - 10.1.4.6 doesn't exist in table
Route 10.25.1.4 through 10.1.4.3
Deleted route: 10.0.0.0 - 10.1.4.3
Route 10.25.1.4 cannot be determined from table
End of program
```

1989 ACM Scholastic Programming Contest
Problem "F"
Twistin' and Turnin'

You have been commissioned by a group of ten year olds to provide them with an encoding program that encodes text resembling English into a dialect known as Twistin' and Turnin', or TNT, as they call it. The rules for encoding the text into TNT follow.

Character Classification

All upper-case and lower-case alphabetic characters ('`A`' through '`Z`' and '`a`' through '`z`'), blank ('` `'), period ('`.`'), colon ('`:`'), and question mark ('`?`') are legal characters. The alphabetic characters `a`, `A`, `e`, `E`, `i`, `I`, `o`, `O`, `u`, and `U` are classified as vowels. All alphabetic characters except vowels are classified as consonants. All characters not mentioned above are illegal.

Elimination Rules

1. Eliminate all illegal characters.
2. Eliminate all leading and trailing blank characters in each line.
3. Replace multiple contiguous blank characters with a single blank character.

Transposition Rules

1. A word is a string of contiguous non-blank legal characters delimited by blank characters and/or by the ends or beginnings of lines.
2. Words that end in a consonant are to have that character moved to the beginning of the word prefixed with the character '`X`'. For example, `Dog` is encoded as `XgDo`, and `Show` is encoded as `XwSho`.
3. Words that begin with a vowel will have that vowel removed and will have the digit (or digits) of the number corresponding to its position in the alphabet placed at the end of the word. For example, `area` is encoded as `real`, and `Ohio4` is encoded as `hio15`. The following table gives the position of each vowel in the alphabet.

Vowel	a, A	e, E	i, I	o, O	u, U
Position	1	5	9	15	21

4. Apply transposition rules 2 and 3 only one time. For example, `ui5*Bc` is encoded as `Xci*B21` ('`5`' and '`*`' are illegal).

Input

The lines of the input file will be at most 80 characters long, but no single word will be longer that 55 characters after applying elimination and transposition rules.

CONTINUED ON REVERSE

Output

1. All encoded lines are to be written with a left margin of 10; that is, 10 blank characters are to appear at the start of each line before any characters of encoded words.
2. Input lines that contain only blank characters and illegal characters (and would thus consist of only blank characters after eliminating illegal characters) are not to appear as output lines.
3. Input lines that consist only of blank characters or that are empty (and thus have no text requiring elimination or transposition) are to cause the termination of writing to the current output line (no more encoded text is to appear on that line), followed by the writing of a totally blank line or null record. That is, a totally blank line will appear in the output. If the current output line is empty (that is, no encoded words have yet been produced), then no totally blank line is to be produced in this case.
4. Output lines are to have a right margin of 65. This means that no part of any encoded word will appear beyond position 65 on any output line. Further, since words are not to be split (continued) across output lines, encoded words that will not fit on the current output line must be word-wrapped as described under Word Wrap Considerations.
5. All encoded words in an output line are to be separated by exactly one blank character. Encoded output is to follow continuously from output line to output line (subject to word wrap considerations) independent of the placement of words in the input.
6. Apply Elimination Rules prior to Transposition Rules.

Word Wrap Considerations

Word wrap is to occur when an encoded word will not totally fit at the end of an output line; that is, it would extend beyond position 65. In this case, no part of this word is to appear on the current output line. Rather, it is to appear in its entirety on the next line immediately following the required initial 10 blank characters.

Example Input

```
     This is a   test for our Twistin! and Turning0 dialect.
It  can  make us use the word wrap feature of  our  program.   In
fact, it will  ****45***7**********  push us down
 a couple of lines.

THE END?
```

Example Output

(The rules at the beginning and end of the output are not to appear in your output. They appear here only for illumination of the output requirements.)

```
1        10                                                  65
+--------+-----------------------------------------------------+
         XsThi Xs9 1 Xttes Xrfo Xru15 XnTwisti Xdn1 XgTurnin
         dialect. Xt9 Xnca make Xs21 se21 the Xdwor Xpwra feature
         Xf15 Xru15 program. Xn9 Xtfac Xt9 Xlwil Xhpus Xs21 Xndow
         1 couple Xf15 lines.

         THE ND?5
+--------+-----------------------------------------------------+
```

1989 ACM Scholastic Programming Contest
Problem "G"
Find A Word

The first 15 lines of the input file for this program will each contain no more than 20 characters (not counting the end-of-line character). Blanks in these lines are significant. Lines containing fewer than 20 characters should be extended to 20 characters by the addition of trailing blanks. These lines form a 15 row, 20 column "Find-A-Word" puzzle matrix.

Each of the remaining lines in the input file will contain a single word, beginning in column 1 of the line. Each word will have no more than 20 characters. Blanks in these lines are to be ignored. The end-of-file is used to mark the end of the input data. The words found in this part of the file are to be located in the puzzle matrix.

Your program is to search the "Find-A-Word" matrix for each of the words found in the second part of the data file. If the word is not found, then an appropriate message should be displayed. If the word is found, the matrix row and column coordinates of each character in the word should be displayed, beginning with the coordinates of the first (leftmost) character in the word. The letters of the word must appear in contiguous cells of the matrix either vertically, horizontally, or diagonally and can be in forward or reverse order. A word may appear only once in the matrix.

CONTINUED ON REVERSE

Example Input

```
B                   S
A FOOTBALL          C
SCOBOL  T           I
E M    I            E
B     D             N
A    E              C
L   -R              E
L  K E  C
  N  T O
 I   UM
LACSAP
    IM
   L O
  E  C
 R
BASEBALL
FOOTBALL
COMPUTER
SCIENCE
EXECUTE
ACM
PASCAL
COBOL
LINK-EDIT
COMPILER
```

Example Output

```
Word                 Where Found
-------------------- ------------------------------------------------------------
BASEBALL             (1,1),(2,1),(3,1),(4,1),(5,1),(6,1),(7,1),(8,1)
FOOTBALL             (2,3),(2,4),(2,5),(2,6),(2,7),(2,8),(2,9),(2,10)
COMPUTER             (14,6),(13,6),(12,6),(11,6),(10,6),(9,6),(8,6),(7,6)
SCIENCE              (1,20),(2,20),(3,20),(4,20),(5,20),(6,20),(7,20)
EXECUTE              NOT FOUND
ACM                  (2,1),(3,2),(4,3)
PASCAL               (11,6),(11,5),(11,4),(11,3),(11,2),(11,1)
COBOL                (3,2),(3,3),(3,4),(3,5),(3,6)
LINK-EDIT            (11,1),(10,2),(9,3),(8,4),(7,5),(6,6),(5,7),(4,8),(3,9)
COMPILER             (8,9),(9,8),(10,7),(11,6),(12,5),(13,4),(14,3),(15,2)
```

1989 ACM Scholastic Programming Contest
Problem "H"
Data Dependency Analysis

One of the steps in analyzing whether an assignment statement can be executed as a vectorized assignment is a data dependency analysis. Your task in this problem is to perform a data dependency analysis on a family of assignment statements involving one-dimensional arrays, linear subscripts, and loops with known integer bounds.

Input for this program consists of a series of specifications for a program's loop and for an assignment statement from within that loop. Each assignment statement involves assignment to a one dimensional array, always called A, and involves the loop control variable, always called `I`. Your task is to determine for each such assignment statement, which of the following conditions apply to it.

a) The assignment will never be executed because of loop conditions.

b) The assignment will be executed exactly once.

c) The assignment will be executed exactly twice.

d) The assignment may be executed as a vector loop.

e) Data dependency forces assignment to be executed in scalar.

All assignment statements will be of the form

```
A[expr]:=expr2
```

where the syntax for a subscript expression may be any of the following:

```
n      n*I      n*I+m      n*I-m
```

where each of `n` and `m` represents a (possibly signed) integer constant in the range –32768 to +32767. There will be no embedded spaces in any expressions. The expression on the right side of the assignment may be any valid Pascal arithmetic expression subject to the requirement that all references to the array A involve subscript expressions restricted to the preceding form. There will be no "trick" expressions.

A statement of this form may be executed as a vector loop if no element of the array A occurs on the left-side of an assignment before its use on the right-side of a subsequent iteration of the assignment statement. If a loop is executed only once or twice, then analysis of vectorization is not required.

The specification `LOOP mm,nn,ll` directs execution of an assignment statement for each value of `I` in the range `mm` to `nn` in increments of `ll`. If `ll` is not provided, then a default value of 1 is used. Each of the integers in the `LOOP` specification will be in the range –32768 and +32767 inclusive.

Output Specification

Your program's output should consist of two lines for each line of input. The first line should echo the input as read, and the second line should report the results of analysis using one of the following phrases:

CONTINUED ON REVERSE

```
Never Executed
Executed Once
Executed Twice
Vector Execution
Scalar Execution
```

Examples of the Analysis

`LOOP 3,9,3: A[I]:=A[I]+A[I-3]`

is not vectorizable because A[3] appears in the first iteration's left side, and the second iteration's right side.

`LOOP 1,11,3: A[I]:=A[I]*2.0+1.798`

is vectorizable because references to `A[I]` occur only in the same iteration in which `A[I]` is changed.

`LOOP 15,1,-1: A[-2*I+31]:=3165.2*A[I+1]`

is not vectorizable because (among other reasons) A[15] depends on the *new* value of A[9], which is set by an earlier iteration of the loop.

Input Format

Each input line takes one of the two following forms:

```
LOOP mm,nn,ll: A[expr]:=expr2
LOOP mm,nn: A[expr]:=expr2
```

Each such line, including the `LOOP` specification and assignment specification, but not including the end-of-line character, is no longer than 80 characters. Trailing blank characters may be present after the assignment specification; these are to be ignored. Your program is to process (independently) as many input data lines as are present. The data terminates with a single line beginning END.

Example Output (Includes Echo of Input)

```
LOOP 1,32767: A[I]:=A[I]+1
Vector Execution
LOOP 1,7: A[-2*I+50]:=A[I-1]+A[I+40]
Scalar Execution
LOOP 2,800,2: A[I]:=-2.03E-17*A[2*I-1]+A[I]
Vector Execution
LOOP 7,1,-1: A[I]:=A[I+6]**3
Scalar Execution
LOOP 2,-90,3: A[I]:=2
Never Executed
LOOP 2,3: A[I]:=A[I]*4.0
Executed Twice
END
```

1989 ACM Scholastic Programming Contest
Addenda to the Problem Statements

Problem B

The perimeter of the island is considered to be part of the island.

Problem F

In the section headed "Output," change specification 1 to read as follows:

1. All encoded lines are to be written with a left margin of 9; that is, 9 blank characters are to appear at the start of each line before any characters of encoded words.

In the section headed "Output," delete the last complete sentence in specification 3. That is, the sentence starting "If the current output line is empty:.." should be deleted.

Problem H

The syntactically valid forms for a subscript expression should read:

n n*I n*I+m I I+m I-m

Change the sentence:

There will be no "trick" expressions.

to read:

There will be no "trick" expressions (e.g. no function invocations).