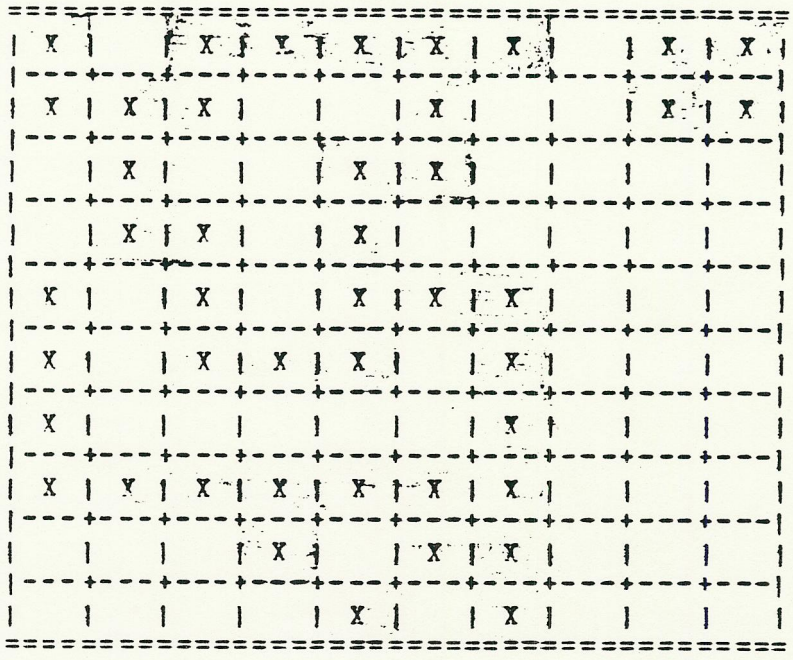
A

Checkerboard Regions

Consider a 10x10 checkerboard, each square of which is colored either black or white. A region of the checkerboard is any collection of contiguous squares. (Two squares are not considered contiguous if they share a corner but do not share an edge.) A region is a hole if all its squares are white and if every edge not totally within the region is either on the checkerboard boundary or is shared by a black square. For example, the checkerboard below contains 8 holes.



(X squares are "black"; empty squares are "white")

You are to write a program that finds the number of holes in such a checkerboard and prints the number of white squares in each hole. For the figure above, your program should print that there are 8 holes, containing 1, 1, 1, 2, 6, 7, 8, and 29 squares. (The holes may be printed in any order, not necessarily in ascending order by number of squares.)

The input consists of ten cards, representing the 1st, 2nd, 3rd, ..., 10th row of squares of the checkerboard. Each card contains ten 1's and 0's in columns 1-10. For a given row, column i represents the color of the square in the i-th column of the checkerboard. Black is represented by 1, white by 0. The input is read from unit 5, and your printer is unit 6. You may assume there are no errors in the input.

B

Manufacturer's Break-Even Analysis

This problem is to calculate the break-even quantity for a given product to be manufactured. The break-even quantity is that point at which the total cost is equal to total revenues. Thus, the breakeven point identifies the minimum quantity of a product that must be sold to make the firm's revenue equal to the total cost of producing that product.

The total revenue (P) is equal to the sales price per unit (P) times the quantity to be produced (Q). The total cost (C) is equal to the fixed operating cost (F) plus the variable cost per unit (V) times the quantity to be produced (Q).

The break-even point is found when total revenue (P) equals or exceeds total cost (C). The profit or loss per unit (U) can be calculated by subtracting the total cost (C) divided by the quantity to be produced, from the sales price per unit (P).

For this problem, you are given the fixed operating cost (F), the variable cost per unit (V), the sales price per unit (P), and the range (N) of units to be considered (i.e., the range of 1 to N units are to be produced). You are to create a table indicating the total cost, total revenue, total profit (or loss) and profit (or loss) per unit for the range indicated. Divide the range into 25 equal steps for this analysis. If N is not divisible by 25, use steps that are the integers closest to (k*N)/25 for k=1,2,...,25.

The input comes from unit 5, in the following format.

Product name	20A1
Fixed operating costs (F)	F10.2
Variable cost per unit (V)	F5.2
Sales price per unit (P)	F5.2
Range (N) of units to consider	I4

Output to be printed on unit 6 is a table with the product name at top and appropriate headings for TOTAL COST, TOTAL REVENUE, TOTAL PROFIT (OR LOSS), PROFIT (OR LOSS) PER UNIT. The table is to contain 25 rows, corresponding to the 25 increments of the range. Print a star (*) next to the row that is closest to the break-even point. If two rows are equidistant from (meaning within one-half cent of) the break-even point, print a star next to both.

C

Interpreter

The following is a BNF description of a very simple programming language.

```
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<var> ::= A | B | C | D | E
<oper> ::= + | - | *
<number> ::= <digit> | <digit> <digit>
<constituent> ::= <number> | <var>
<expression> ::= <constituent> | <constituent> <oper> <constituent>
<statement> ::= <var> = <expression> | STOP
```

Note that the following are valid statements (with the obvious meanings) in the language,

```
A=4
B=38
C=A+B
D=62*B
STOP
```

while the following are not valid statements in the language.

```
K=31        (K is not a valid variable)
B=962       (number must be no more than two digits)
C=-A        (unary minus not defined)
D=A+B*C     (only one operator allowed)
A = B + C   (no spaces allowed)
```

You are to write an interpreter for this language. The interpreter should accept valid statements, down through the first STOP statement. When the STOP is encountered, it should print each variable assigned a value, as well as its current value.

For example, given the following statements,

```
A=5
D=6*A
C=D+60
STOP
```

the program should print

```
A=      5
C=     90
D=     30
```

You are not to execute a statement if it requires the value of a variable that has not yet been assigned a value. Your program will be given only valid statements in this language. You may assume that a STOP statement is present in the input. You may also assume that the magnitude of each variable in the interpretation is not more than nine digits, and can be contained in a single integer

D

Gas Mileage

An auto manufacturer is conducting a mileage experiment in which a number of vehicles are to be driven in normal use. Each driver will be assigned a car and will record information about gasoline consumption, operating conditions, etc. In particular, each time the driver buys gasoline, the driver records the odometer reading, the number of gallons purchased, whether or not the tank was filled, and the purchase price. You are to write a program to tabulate gasoline consumption information for all cars of the experiment.

Input to the program is from cards read from unit 5. The input format is as follows.

cols. 4-6	car number (integer between 0 and 999, inclusive)
cols. 8-13	odometer reading when gasoline purchased (implied decimal point between cols. 12 and 13)
cols. 15-17	gallons purchased (implied decimal point between cols. 16 and 17)
cols. 19-22	cost in dollars (implied decimal point between columns 20 and 21)
col. 24	'F' if tank was filled; ' ' otherwise

The input contains one or more cards of this format. Cards are grouped in packets; each packet contains the description of one car number, and each car is described in only one packet. Within a packet cards are in ascending order by odometer reading. Input is terminated by a card containing a negative car number. You may assume that the input is in the format specified and it contains no errors. The first card will give only an initial odometer reading and the indication 'F', denoting that the tank was full at that point. You may further assume that no odometer resets to zero during the experiment.

Output for each car in the experiment should be as follows.

CAR NUMBER xxx

ODOMETER	GALLONS PURCHASED	FILL UP?	COST	MPG	
xxxxx.x		YES			INITIAL
xxxxx.x	xx.x	zzz	xx.xx	xx.x	
	. . .	etc.			
xxxxx.x	xx.x	zzz	xx.xx	xx.x	
xxxxx.x	xxx.x		xxxx.xx	xx.x	FINAL

where x's indicate positions of digits and zzz represents 'YES' or 'NO '. You may assume that the number of digits specified in each of the above fields is sufficient. Note that the MPG column should represent mileage between fill up's, and should be printed only when this figure can be calculated. The FINAL line contains the number of miles driven, the number of gallons of gasoline purchased, the total cost of the gasoline, and the miles per gallon computed from the initial odometer to the most recent fill up. Spacing of your

E

Team Standings

You will be given 30 records on input unit 5. The first 20 are in the following format.

cols. 1-12 team-A name, alphanumeric (left justified)
cols. 13-14 team-A score, integer (right justified)
cols. 15-20 blanks, ignored
cols. 21-32 team-B name, alphanumeric (left justified)
cols. 33-34 team-B score, integer (right justified)
cols. 35-80 blanks, ignored

The next 10 records have the following format.

cols. 1-12 team name, alphanumeric (left justified)
cols. 13-20 blanks, ignored
cols. 21-32 team name, alphanumeric (left justified)

The first 20 records constitute the records of 20 or fewer teams in 20 different games, one game per record. For example, a record with contents
FLORIDA 17 AUBURN 14
would indicate that FLORIDA defeated AUBURN by a score of 17 to 14. The winning team will not necessarily be listed first. There are no ties, and no team plays the same team more than once.

The last 10 records correspond to queries about the "facts" presented in the first 20 records. Each record will contain two team names. You are to determine whether
(a) one team has defeated the other,
(b) one team has defeated the other indirectly, meaning that it has defeated a team that has defeated the other, or that it has defeated a team that has defeated a team that has defeated the other, etc.
(c) each team has defeated the other indirectly, or
(d) neither team has defeated the other either directly or indirectly.

In these four cases, you are to print a line on unit 6 for each input query, as appropriate, resembling one of the following formats.
(a) team-A DEFEATED team-B
(b) team-A DEFEATED team-B INDIRECTLY
(c) team-A AND team-B HAVE DEFEATED EACH OTHER INDIRECTLY
(d) team-A AND team-B ARE NOT COMPARABLE

F

Editing Numbers

The purpose of this problem is to edit an integer number using a "mask" to produce output in a format with commas, a decimal point, etc. suitable for printing. Write a program to read a collection of records in the following format:

cols. 1-10 integer number, right justified; any leading nonsignificant columns may be either spaces or zeros. You may assume the value is small enough to fit into a single INTEGER variable.
cols. 11-25 mask (ending with a space)
cols. 26-80 spaces (ignored)

The mask will be left justified in the field and will consist of the characters 'Z', '9', ',', '-', and '.'. Characters in the mask are processed from left to right against the digits. Each Z or 9 in the mask applies to one digit in the input integer; the number of digits processed in the integer is equal to the total number of Z and 9 characters in the mask. If the mask has n Z and 9 characters, then the rightmost n digits of the integer number are edited. The following rules govern processing.

1. Z characters are to be replaced with either a digit from the integer number (if the digit is not a leading zero or space) or with a space (if the digit is a leading zero or space).

2. 9 characters will be replaced with a digit from the integer number, even if the digit would ordinarily be considered a leading zero. All zeros in the integer number processed after a 9 character in the mask are considered to be non-leading zeros.

3. The ',' and '.' characters are to be replaced with a space if only leading zeros have been processed; otherwise, they will appear in the output in the position they appear in the mask.

4. The '-' sign, if present, must appear at the end of the mask and should be replaced with a space if the number is positive; otherwise it should appear in the position it appears in the mask.

Each card in the deck should be read and processed as described above. Each edited field should be printed in a table similar to the following, although the spacing of individual items may differ slightly. The input comes from unit 5; the output goes to unit 6. A totally blank input card marks the end of the input data.

Examples:

Number	Mask	Edited Value
0234567	ZZZ,ZZ9.99	2,345.67
-1234567	Z,999.999-	1,234.567-
123	ZZ,Z99.99	01.23
1234567	Z,999,999-	1,234,567
1234567	Z,999	4,567