1985 ACM PROGRAMMING CONTEST

Subset Words

Program File:	SUBSET.PAS	---> Pascal Version
	SUBSET.FOR	---> FORTRAN Version
Executable File:	SUBSET.EXE	
Input Data File:	C:SUBSET.DAT	
Output File:	C:SUBSET.OUT	

In this problem you are asked to determine if two words come from the same base alphabet. If one word is a rearrangement of the other, the two words are called "anagrams", like "rose" and "sore". This problem asks you to determine if two words use the same letters, but do not necessarily use equal numbers of them. For example, "curse" and "rescue" come from the same base alphabet, but "cure" does not, since its base alphabet does not contain the "s" that both "curse" and "rescue" use.

Your program is to REPEATEDLY read pairs of words from a file. The pairs are presented one per line and are left justified in the first twenty positions of the line (any remaining positions are filled with blanks). Your program should output to a file both words and a message indicating that they use the same letters or different ones. A pair of words, "LAST" and "PAIR" will indicate the end of the data. The last pair of words is not to be processed.

1985
ACM PROGRAMMING CONTEST

The Wizard's Square

Program File:	WIZARD.PAS	---> Pascal Version
	WIZARD.FOR	---> FORTRAN Version
Executable File:	WIZARD.EXE	
Input Data File:	C:WIZARD.DAT	
Output File:	C:WIZARD.OUT	

A peasant was trying to solve the problem of placing the integers from one to sixteen into a four by four matrix, such that if the products of the numbers in each row and the products of the numbers in each column were computed, and then those eight products were added together, the sum would be as large as possible. The peasant had drawn his matrix in the sand with a stick, and had placed the sixteen numbers in the matrix.

The old wizard was riding by on his dragon, and saw the matrix drawn by the peasant. The wizard remarked, "Your matrix is good, but the largest sum can be achieved by rearranging these four numbers." He dismounted and rubbed out four of the numbers in the matrix. However, before he could replace the numbers in the matrix, he gasped and fell dead of old wizard's syndrome.

Write a program that will reconstruct the matrix that the wizard was about to complete. The input will come from a file and will consist of a single line containing sixteen integers, each right justified in a three column field. These values represent the matrix at the time of the wizard's death. The first four values are the first row, the next four the second row, etc. Four of the input values will be zeroes, representing the values that the wizard had erased. The other values will be in the range 1 to 16, and are in the positions chosen by the peasant.

Your program shall output to a file the reconstucted matrix as a square, with four rows of four values each. This matrix will be followed by the value of the matrix, which is the sum of the row and column products.

1985 ACM PROGRAMMING CONTEST

Pattern Matching

```
Program File:      PATTERN.PAS       ---> Pascal Version
                   PATTERN.FOR       ---> FORTRAN Version
Executable File:   PATTERN.EXE
Input Data File:   C:PATTERN.DAT
Output File:       C:PATTERN.OUT
```

A ten by ten array of characters contains 5 "X"s, 5 "O"s, and 90 periods (see examples below). The 5 "X"s are all connected, either orthogonally or diagonally; similarly for the 5 "O"s. It is desired to determine whether the pattern of "X"s matches the pattern of "O"s, by which we mean that one pattern can be exactly superimposed on the other. Only translations and rotations are allowed; relections are not allowed.

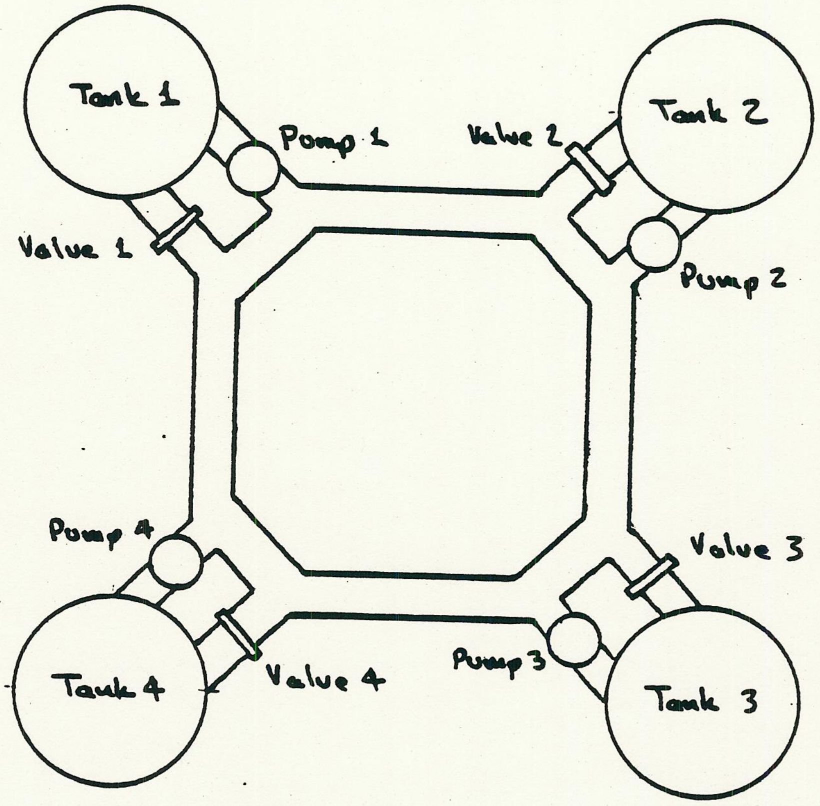
There will be exactly four data sets in a file to process. Each data set consists of ten lines of ten characters each (periods, "X"s, and "O"s). The letters are uppercase, and there are no intervening blanks. The output for the program shall be written to a file. For each data set, your program should skip a line and then write the data set as ten lines of characters, but with one space between adjacent characters (exactly as shown below). The your program should determine if the two patterns match, and write either "Patterns match" or "Patterns do not match", as appropriate, directly under the array.

Below are examples of output from two data sets, correctly written, with the correct output message below.

```
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . X . . .
. . . . . . X X . .
. . . . . . X . . .
. . O . . . X . . .
O O O O . . . . . .
. . . . . . . . . .
. . . . . . . . . .
Patterns do not match

. . . . . . . . . .
. . . . . X . . . .
. . . . . . X . . .
. . . . . . . X . .
. . . . O O X X . .
. . . O . O . . . .
. . O . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
Patterns match
```

Each command is presented as the first sixteen characters on an input line, each character being either 0 or 1. Following the command is a blank, and then an integer specifying when the command is issued, measured in seconds from the start of the simulation (which is time 0). The commands appear in strictly increasing order of time, one per line. It is not possible for more than one command to be given at the same clock time.

Your simulation should process all input commands, and then continue running until the tank sensors cause all pumps to be turned off. Then output to a file the clock time (number of seconds elapsed since the beginning of the simulation) and the current amount of liquid in each tank, accurate to one decimal place. Label each value appropriately.



1985 ACM PROGRAMMING CONTEST

Chemical Plant Control

Program File:	PUMPS.PAS	---> Pascal Version
	PUMPS.FOR	---> FORTRAN Version
Executable File:	PUMPS.EXE	
Input Data File:	C:PUMPS.DAT	
Output File:	C:PUMPS.OUT	

A chemical plant has four tanks, connected by pipes as shown in the diagram. At each tank is a valve that controls flow of liquid into the tank, and a pump that controls flow out. A process control computer controls the valves and pumps, receiving its input from an operator console and from sensors in the tanks. You are to write a program that simulates the transfer of liquids among the tanks, as described below.

Initially, assume all valves are closed and all pumps are off. Assume also that all pipes are filled, so that liquid pumped out of one tank instantaneously begins to flow into another tank. If more that one valve is open, the flow is evenly divided among the tanks, It is possible for more than one pump to be on at the same time. Liquid may enter a tank only through its valve, and leave only through its pump.

The input data for this problem will be read from a file. The first four lines of the input data will give the total capacity and initial contents (in liters) of each tank (tank 1 on the first line, etc.). Each line will contain two real numbers separated by a blank (capacity, then contents). The fifth input line will contain a single real value, the pump capacity in liters per second. All pumps have the same capacity. You may assume that the tank capacities and contents are less than or equal to 10,000, that the pump capacity is no less than 1 liter/second, and that all of these numbers have no fractional parts.

Once each second the computer performs the following steps. It gets a command from the operator console (described below), and performs the command. It then checks the tank sensors. If any tank has been filled to 95% (or more) of its capacity, its valve is closed. If any tank have been emptied to 5% (or less) of it capacity, its pump is turned off. Assume these precautions prevent tanks from ever being emptied or filled, except possibly in their initial state. To avoid overstressing the pumps, if all valves are closed, all pumps are turned off.

The computer receives commands that are 16-bit words. Each 0 bit is ignored, and each 1 bit specifies an action. The first four bits represent the action of turning on pump 1 through 4, the next four bits represent turning off pump 1 through 4, the next four bits opening valve 1 through 4 and the last four bits closing valves 1 through 4. The computer must ignore conflicting actions; for example, if bits 1 and 5 are both ones (meaning turn pump 1 on and turn pump 1 off) then both actions are ignored and pump 1 is unaffected. Similar conflicting valve actions are also ignored.

1985 ACM PROGRAMMING CONTEST

Inferred Handicapping

Program File:	HANDICAP.PAS	---> Pascal Version
	HANDICAP.FOR	---> FORTRAN Version
Executable File:	HANDICAP.EXE	
Input Data File:	C:HANDICAP.DAT	
Output File:	C:HANDICAP.OUT	

Due to a strike by Local 2 of the Amalgamated League Secretaries and Plaster Casters Union, the manager of a nearby bowling alley, Peter Piper's Pin Palace, needs some programming in a hurry to automate the calculation of averages and handicaps for the leagues that bowl there. The parameters in each league's handicapping formula are not available, but some old status sheets for each league have been found and must be used to infer what the parameters might be. Naturally, none of the league secretaries will tell what values his or her league uses.

The handicap calculation algorithm is known to be the same for all leagues and goes as follows:

1) The bowler's (per game) average score, truncated to an integer (between 0 and 300) is subtracted from a parameter called PAR.
2) If the difference is negative, the handicap is zero. Otherwise, a percentage given by the parameter FCT is taken and the result truncated to an integral value for the handicap.

Symbolically:

Handicap = floor (max(0, PAR - average) * FCT/100)

It is known to be a national policy that the parameters are integers with FCT in the range 50 to 100 and PAR in the range 150 to 300. The data file contains a record for each league and contains the following information right justified in each field:

Columns	Contents
1-3	League Number
5-6	Number of Average-Handicap Pairs in Record
8-10	Average 1
12-14	Handicap 1
16-18	Average 2 (if present)
20-22	Handicap 2 (if present)
.	.
.	.
64-66	Average 8 (if present)
68-70	Handicap 8 (if present)

You may assume that all other characters in the line are blanks.

For each record in the file until League Number = 0, the program is to output to a file in a readable format the League Number and all FCT-PAR pairs which could be used to give the Average-Handicap pairs given for that league. If not such pairs exist, print a notice that at least one given handicap is wrong and print all the FCT-PAR pairs which could be used to give all but one of the Average-Handicap pairs on the league's data record. If no FCT-PAR pairs are consistent with fewer than two errors in the input data, so indicate.

OUTPUT SPECIFICATION

For each data set, your program should write to a file the formatted text centered within an 80 column line. The first line written to the file is to be a border line created from the user selected character followed by the remaining lines of formatted text (beginning and ending with the border character) and ending with another border line at the end of the text. Each data set should begin at top of a new page.

EXAMPLE

A Sample Data Set:

```
19
*
6
This is some sample text that has been formatted
by
this program for display.  The specified
line length was 19 characters with the text to be
surrounded by a border consisting
of '*'.
```

Sample Formatted Text (centered in 80 columns):

```
                              *********************
                              *This is some sample*
                              *text that has been *
                              *formatted by this  *
                              *program for        *
                              *display.  The      *
                              *specified line     *
                              *length was 19      *
                              *characters with the*
                              *text to be         *
                              *surrounded by a    *
                              *border consisting  *
                              *of '*'.            *
                              *********************
```

1985 ACM PROGRAMMING CONTEST

SCREEN FORMATTING

Program File:	SCREEN.PAS	---> Pascal Version
	SCREEN.FOR	---> FORTRAN Version
Executable File:	SCREEN.EXE	
Input Data File:	C:SCREEN.DAT	
Output File:	C:SCREEN.OUT	

Write a program that will read text from a file and reformat the text for display. The text is to be centered in the display area. It should be surrounded by a border constructed from a character specified in the input data.

For each data set, the text should be displayed as follows:

1. For each data set, the output is to begin on a new page and should be centered in a display area 80 columns wide. Output for each data set will reside on a single page.

2. The line length for displayed text will be specified in the input data and will be within the limits of 1 - 78 characters.

3. The text should be formatted to the specified line length and should be printed in the display area completely surrounded by a border as specified by the input data.

It can be assumed that for each data set, the text contains no words that are longer than the selected line length and once formatted can be printed on a single line printer page. The delimiters for words in the text are spaces, end-of-line, beginning-of-line, and end-of-file. Each formatted line of text is to contain the maximum possible number of words that create a line less than or equal to the selected line length.

INPUT SPECIFICATION

Multiple data sets are contained in a file arranged as follows:

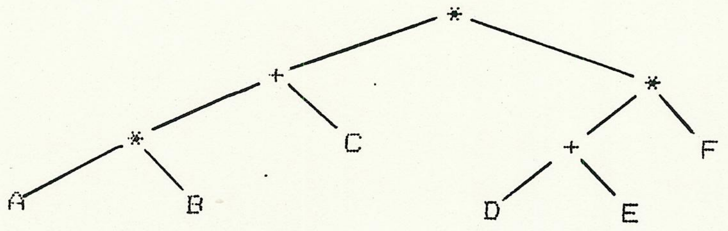
```
     Record #                Value

        1          Line length for displayed text (1-78)
        2          Character to be used for the border
        3          The number of text lines following (1-50)
        4    --+
        .      |
        .      +-The text to be formatted and printed.
        .      |    (contains no adjacent blanks)
        n    --+
```

1985 ACM PROGRAMMING CONTEST

Printing Words from Binary Pattern Trees

Program File:	BINWORDS.PAS	---> Pascal Version
	BINWORDS.FOR	---> FORTRAN Version
Executable File:	BINWORDS.EXE	
Input Data File:	C:BINWORDS.DAT	
Output File:	C:BINWORDS.OUT	

In a pattern tree, leaf nodes contain letters as data and have no children. Interior nodes have either a "+" or a "*" as data and have two children -- a left subtree and a right subtree. A "+" means alternation -- either the string from the left subtree or the string from the right subtree may be chosen here. A "*" means concatenation -- the string form the left subtree must be concatenated with the string from the right subtree. For example:



represents the following words:

```
ABDF
ABEF
CDF
CEF
```

You are to write a program which reads in binary pattern trees from a file and then outputs to a file all words represented in each pattern tree. Input data will occur in sets, with one set representing a pattern tree. There will be a maximum of 50 nodes in a pattern tree. The first record in the set contains a number of nodes in the tree, N, in right justified in columns 1-2. The next N records contain the following information about the nodes:

Column	Contents
1-2	number of the node (the root is always node #1; other nodes may have any number, in any order)
4	data at the node (either an alphabetic character or a "+" or a "*")
6-7	number of the left child node (0 = none)
9-10	number of the right child node (0 = none)

All values are right justified in the indicated fields.

For each tree, output to the file all words represented by that tree, one word per line. Continue processing sets of pattern tree data until a trailer record with a negative value for the number of nodes in encountered.