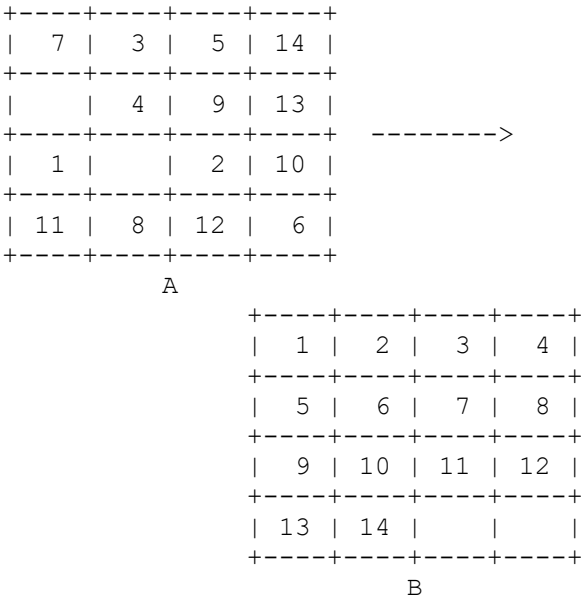
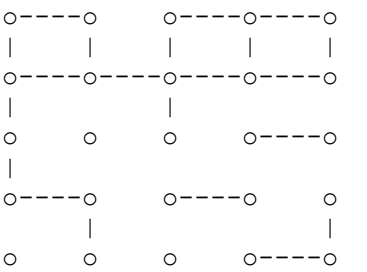
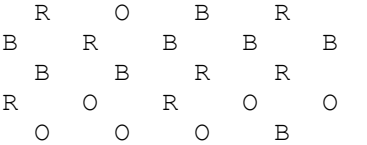
PROBLEM 1.
Program the process transforming figure arrangement in A into that in B. A figure in a cell can be transferred into any one of neighbouring empty cells; in the case of such transformations the cell previously occupied by the figure becomes empty.



PROBLEM 2.
The game of "BOXES" involves players joining dots in a gird one at a time. The player who completes a box scores a point and is entitled to another turn. The end of the game is when the grid has been completed with the winner naturally being the one with the more points. Only vertical and horizontal lines filling the space between two points are allowed.
The example below shows a game between Red and Blue which is half completed:



This can be represented with R & B showing lines filled by Red and Blue with O indicating a 'vacant' line:



```
B    O    O    O    O
  B    O    R    O
O    R    O    O    R
  O    O    O    B
```

This is then condensed to produce the matrix shown below:

```
ROBR
BRBBB
BBRR
ROROO
OOOB
BOOOO
BORO
OROOR
OOOB
```

1. Write a program which scans a matrix of the type shown above (which will ALWAYS represented a 5 x 5 point grid) and determines the number of 3- sided boxes (of ANY orientation). Data should be read from a single file as a series of 9r lines representing r games which is terminated by the word END starting a separate line. Your program should output a one line message for each matrix in the data:
   MATRIX r contained x 3-sided box(es).
2. Modify your answer to Part 1 to continue a single game on behalf of Blue. Complete as many boxes as the single move allows (bearing in mind that a complete box means another move). The final move is irrelevant but should NOT result in a 3-sided box unless forced to do so.
   Your program should return:
   x boxes have been completed. Final move = L, C where L & C are the Line and Column representing the character in the matrix and x should be the number NEW boxes which have been formed, but NOT the total number of complete boxes in the grid.

*** PROBLEM 3.

There are N books and two readers, A and B, wanting to read these books. Nonnegative integers A[I] and B[I] are given such as reader A (or B) needs A[I] (or B[I], respectively) hours to read book I, 1<=I<=N. Both the readers begin reading at time 0. At any time each reader is allowed to read at most one book and both readers cannot read the same book.

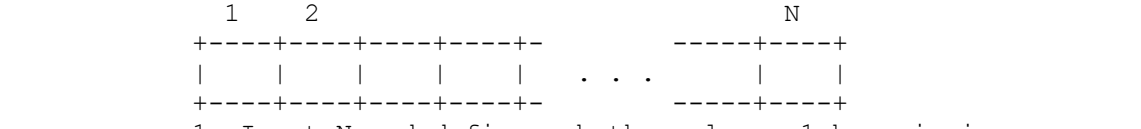
Integer K, 2<=K<=N, is given and the books are supposed to be numbered in such a way that no reader can start reading book J, 2<=J<=K, until book J-1 is read by both the readers.

The order of reading the other books is immaterial for each reader and may be arbitrary.

Preemptions are allowed in the process of reading any book by any reader. It means that this process may be interrupted at any integer time and be resumed lately starting from the point of interruption. In between interruption and resumption of the process of reading the book a reader may read any other book he has not completed and has the right to read it.

IT IS NECESSARY:
1. To organize inputting the data in the form:
   < ENTER N --> >
   < ENTER K --> >
   < ENTER: >
   < A[1] --> > < B[1] --> >
   < A[2] --> > < B[2] --> >
   ....
   < A[N] --> > < B[N] --> >
2. To find the largest possible time T before which all the books cannot be read by both the readers; to output calculated value of T.
3. To build a schedule of reading the books by each reader which meets all the restrictions listed above and under which the process of reading all the books terminates at time T. The schedule for each reader is to be written in the form.
   < SCHEDULE FOR READER A (or B) >
   < Book > < Start > < Finish >
     ....
     ....
   In the tables of the above form all the time intervals within which reader A (or B) is reading a book and the number of this book should be mentioned.
4. Output the number of preemptions of each reader. Try to reduce the number of preemptions for each reader.

PROBLEM 4.
It's given integer number K.
A strip of paper is divided into N cells (K<=N<=40). Two players choose and cross out K empty adjacent cells one by one. The winner is the one who has made the last move.



1. Input N and define, whether player 1 has winning strategy (i.e. whether he can win under the best following moves of player 2). Print message "Player 1 has winning strategy" or "Player 1 doesn't have winning strategy".
2. Define for given N, if player 1 has winning strategy, if his 1st move is entered into the computer from keyboard.
3. Make the game for given (paragraph 1 and 2) N and player's 1 first move. Programme plays for player 2. Moves of player 1 are entered from the keyboard. Move is given by index of cell L (1<=L<=N-K+1). Cells from L till L+K-1 are crossed out while doing this. After each move current position of the game is printed out in the form of:
   1 2* 3* ... N
   Index number is printed upside, crossed out cells are marked by symbols '*'.
   You must print 'Victory of Player 1 (Player 2)' when the game is over. Entering N and K print message 'N>' and

'K>'.
Entering move print 'Move of Player 1>'.
Foresee the control of correctness of input data.

PROBLEM 5.
[PROLAN/M]
Suppose that the NePhihhan hardware company has developed a new RISC micro-processor capable to handle a single data type - string of characters - and to perform a single operation on them-context-sensitive replacement (searching a given substring in a string and replacing it by another substring). Two memory areas are used, one of each contains the program (a list of discriptions of the possible replacements), while the other one (we will call it R; its size is supposed to be virtually unlimited) is used to store the input data, the intermediary results and the final output.
The programs for the processor are written in a language which we will call PROLAN/M. Before we describe it formally, we will give an example of a program:
(aa,b) (ba,a) (bc,a) (c,start) (d,) (b,finish) (,)
When executed string abcabcd
as input, the programs yields the string finish
as a final value, while the contents of R goes through the values
abcabcd, aaabcd, babcd, abcd, aad, db, b, finish
successively.
Now to the formal description of the syntax of PROLAN/M (we use "::=" to denote "is defined as" and ":" to denote "or"):
<'PROLAN/M'-program> ::= <substitut.sequence>(,)
<substitut.sequence> ::= <substitution>:
::= <substitut.sequence><substitution>
<substitution> ::= (<left part>,<right part>)
<left-hand part> ::= <string>
<right-hand part> ::= <string>:<empty>
<string> ::= <string symbol>:<string><string symbol>
<empty string> ::=
<string symbol> ::= <any ASCII character except ','
, ')' >
After the input string has been loaded into R, the program is executed in the following way: the processor looks for the first <substitution> in the <substitut.sequence> for which the <left-hand part> is a substring of the string in R. If the search is successful, the <right-hand part> of the same <substitution> replaces the corresponding substring in R (the leftmost one if not unique). This procedure is then repeated from the beginning with the new R-value until mo <left-hand part> in the <'PROLAN/M'-program> is found as a substring in a current value R, which is then considered to be the final result, and the execution is aborted.

Problem 1.
Write and debug a PROLAN/M program that converts a string

of the type
<nat.nr1>+<nat.nr2>=?
(<nat.nr1> and <nat.nr2> are sequences of decimal digits representing natural numbers) to a string of the type
<nat.nr1>+<nat.nr2>=<nat.nr3>
containing a mathematically correct statement (<nat.nr1> and <nat.nr2> are the same). For example, the string
1990+123=?
should be transformed into
1990+123=2113
at the end of the execution. Store your program in a file named SUM.PRM.

Problem 2.
Write a PROLAN/M debugger. It should able to do the following:
(a) request the name of a text file containing the PROLAN/M program;
(b) request the initial contents of R;
(c) perform the transformations of the input string according to the program in the file;
(d) display the result on the screen;
(e) it is desirable to enable a tracing mode.

Your grade for P.1 will depend on the number of <substitutions>s in the <substitution sequence>, as well as on the speed at which the tests which will be given by the jury will be performed. Therefore you may wish to hand in two versions of the program, each of which will be better at satisfying a different criterion.
The program from P.1 will be tested using a programming system designed by the jury for this special purpose.
Your grade for P.2 will depend on passing the jury's test and on your having implemented all the subproblems.

PROBLEM 6.
Given integers a and n (n<100). Suppose an imaginary programming language containing an assignment statement and a multiplication operator. Write a program that generates a text in that language for computation of b=a^n, with minimal number of multiplications. An example of generated text for n=13, where each pair of brackets {} contains comments, is presented below:

```
X1:=a;    {=a}
X2:=X1*X1;     {=a^2}
X3:=X2*X2;     {=a^4}
X4:=X3*X1;     {=a^5}
X5:=X3*X3;     {=a^8}
X6:=X5*X4;     {=a^13}
b:=X6;
```