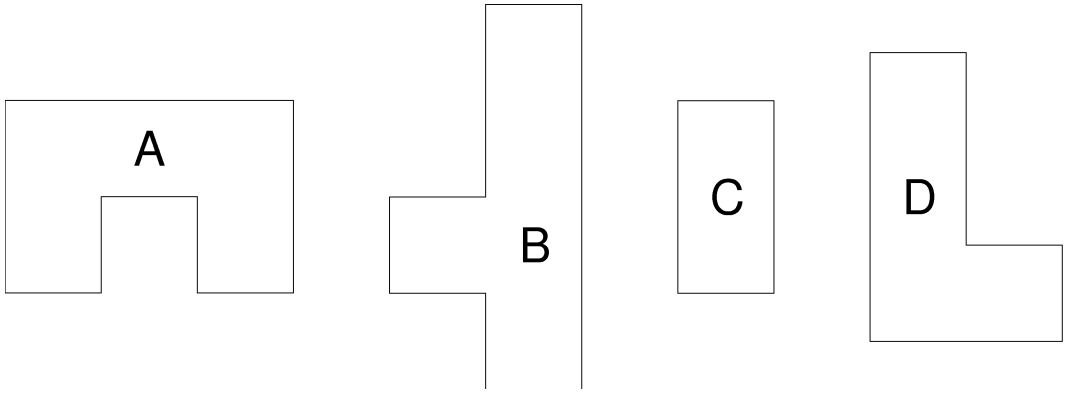
1995 ACM MID-CENTRAL REGIONAL PROGRAMMING CONTEST

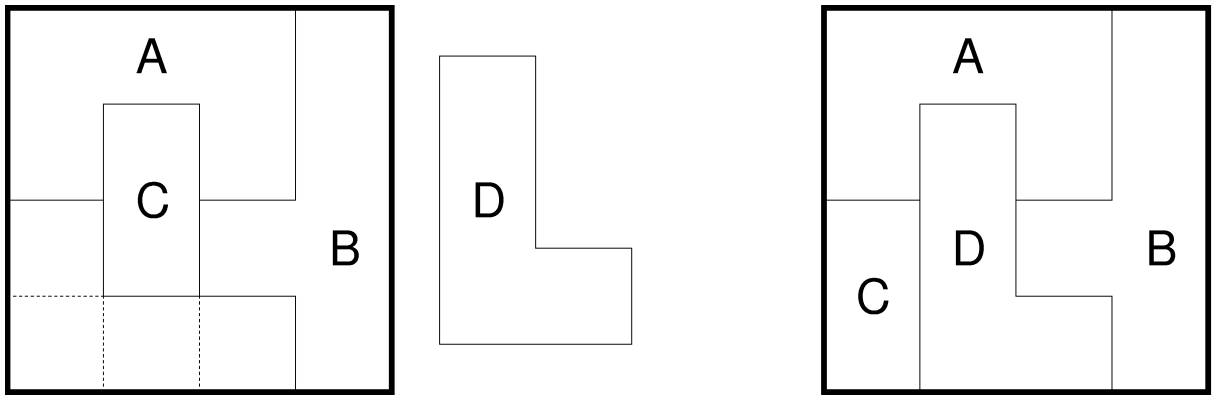
Problem #3 - A Puzzling Problem

Source File: **puzzle.{c|cpp|pas}**
Input File: **puzzle.in**
Output File: **puzzle.out**

The goal of this problem is to write a program which will take from 1 to 5 puzzle pieces such as those shown below and arrange them, if possible, to form a square. An example set of pieces is shown here.



The pieces cannot be rotated or flipped from their original orientation in an attempt to form a square from the set. All of the pieces must be used to form the square. There may be more than one possible solution for a set of pieces, and not every arrangement will work even with a set for which a solution can be found. Examples using the above set of pieces are shown here.



The input file for this program contains several puzzles (i.e. sets of puzzle pieces) to be solved. The first line of the file is the number of pieces in the first puzzle. Each piece is then specified by listing a single line with two integers, the number of rows and columns in the piece, followed by one or more lines which specify the shape of the piece. The shape specification consists of '0' and '1' characters, with the '1' characters indicating the solid shape of the puzzle (the '0' characters are merely placeholders). For example, piece 'A' above would be specified as follows:

```
2 3
111
101
```

The pieces should be numbered by the order they are encountered in the puzzle. That is, the first piece in a puzzle is piece #1, the next is piece #2, etc. All pieces may be assumed to be valid and no larger than 4 rows by 4 columns.

The line following the final line of the last piece contains the number of pieces in the next puzzle, again followed by the puzzle pieces and so on. The end of the input file is indicated by a zero in place of the number of puzzle pieces.

Your program should report a solution, if one is possible, in the format shown by the examples below. A 4-row by 4-column square should be created, with each piece occupying its location in the solution. The solid portions of piece #1 should be replaced with '1' characters, of piece #2 with '2' characters, etc. The solutions for each puzzle should be separated by a single blank line. For puzzles which have no possible solution simply report "No solution possible".

A sample input file is shown here:

```
4
2 3
111
101
4 2
01
01
11
01
2 1
1
1
3 2
10
10
11
4
```

```
1 4
1111
1 4
1111
1 4
1111
2 3
111
001
5
2 2
11
11
2 3
111
100
3 2
11
01
01
1 3
111
1 1
1
0
```

The following output file should be produced from the above sample input:

```
1112
1412
3422
3442

No solution possible

1133
1153
2223
2444
```