ECE30018 Problem Solving Studio, Fall 2023

P6. Polyomino Puzzle

| Submission due: 1:00 PM, 31 Oct Tue

Polyomino Puzzle

A polyomino is a rectilinear shape obtained by combining one or more 1x1 square blocks. For instance, Figure 1 shows four polyominos with different combinations of such blocks. Suppose that we are playing with a puzzle to form a square with given polyominos. In this puzzle, each polyomino is not allowed to rotate, and all polyominos are required to be used to form a square. Figure 2 shows how we can achieve a square by arranging the four polyominos in Figure 1.

Write a program that find a solution of this puzzle for given polyominos.

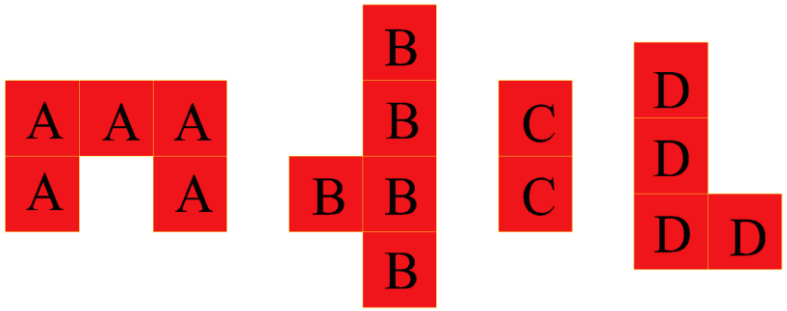

Figure 1. Four polyominos

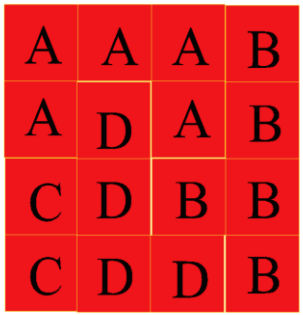

Figure 2. A square composed of the four polyominos in Figure 1.

(Continued)

Requirements

Input Data

- The first line from the standard input has an integer n, the number of polyominos in this puzzle for $1 \leq n \leq 5$. The following lines define the shapes of n polyominos. These polyominos in sequence have IDs from 1 to n, respectively.
- For each polyomino n, $1 \leq n \leq 5$, the first line gives two numbers h and w which represent the height and the width of a polyominos respectively, for $1 \leq h \leq 4$ and $1 \leq w \leq 4$. Then, h lines follow, each of which contains a number consisting of w binary digits, which defines the shape of a polyomino. A binary digit is 1 if and only if the polyomino has a square block at the corresponding position.

Output Data

- Print out an array that represents the arrangement of the given polyominos to the standard output. The array should be a square, that is, the number of rows should be the same as that of the columns. A square block in the array should be specified by the ID of the polyomino at the corresponding position. If it is impossible to form a square array, then print "No solution possible" as the output.
- Your program must return the result within 3.0 seconds

Test Case Example

Input data

```
4
2 3
1 1 1
1 0 1
4 2
0 1
0 1
1 1
0 1
2 1
1
1
3 2
1 0
1 0
1 1
```

Output data

```
1 1 1 2
1 4 1 2
3 4 2 2
3 4 4 2
```