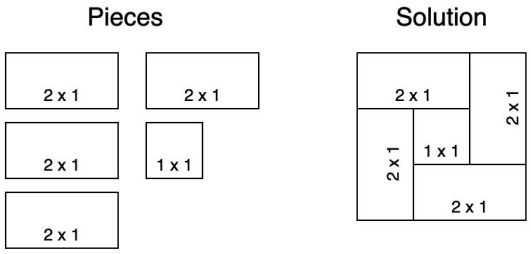
New Year Puzzle

Overview

Your friend bought you a present for the New Year, it's a puzzle! The puzzle consists of a number of wooden rectangular pieces of varying lengths and widths and a board. The goal is to position the wooden pieces on the board in a way such that all of the pieces will fit.

For example, suppose the puzzle has only five wooden pieces: four of which have dimension (2 x 1) and one of which has dimension (1 x 1). The board on which you are trying to fit all of your pieces has dimension (3 x 3). There are many solutions to this example. One such solution is shown in the figure on the right.



Unfortunately, the puzzle your friend has given you is not so easy. You spend hours and hours trying to solve it, but you just can't get the pieces to fit correctly. Finally, you decide to write a program to do it faster. Luckily, your friend is also a programmer and has written some code to solve simpler five piece puzzles. Your friend shares this code with you, but warns you that it contains a few bugs.

Since you know your friend is likely to give you another rectangular wooden piece puzzle next year with different piece sizes and a different board size, you want to design your program to be able to solve puzzles of variable number of pieces and board size.

Input

Your program should take the path to a text file as its only argument. The first line of the text file will contain the width and length of the puzzle board comma-separated. Each subsequent line of the text file will contain a positive integer piece id, piece width, and piece length also comma-separated. All widths and lengths will be positive integers. Your program can assume that the input text file is properly formatted and that the amount of pieces in the puzzle will not exceed 12 pieces.

Output

Given the input, your program should determine and output a corresponding message as to whether the pieces collectively fit perfectly on the given board dimensions or not.

Additionally, if the pieces fit perfectly, output the position and orientation of each piece (identified by the piece's id) in a solved solution.

- Taking the top left corner of the board as the origin, (0,0), the position of each piece on the board is defined as the x, y coordinates of the top left corner of the piece.
- The orientation of each piece can either be V (vertical) or H (horizontal). A vertical orientation indicates that the longer side of the piece is parallel to the y-axis, whereas a horizontal orientation indicates the longer side is parallel to the x-axis.

- Note: For square pieces, vertical and horizontal orientations are equivalent and therefore interchangeable.

Instructions

Attached with these instructions is a program called ***solve.py*** which contains code written to solve puzzles with exactly five pieces. Take some time to read through the code and understand all of the components. This program takes input as defined above, a text file containing board and piece dimensions. The code, however, is not perfect. It is not well-documented and some of the methods contain bugs. Also, two methods in the **PuzzleBoard** class are partially complete.

Next, implement the **solve** function, which should recursively solve the puzzle for a variable board size and up to 12 pieces, using **solve5** as a reference. You are free to add/change methods in the classes, create helper functions, and add comments.

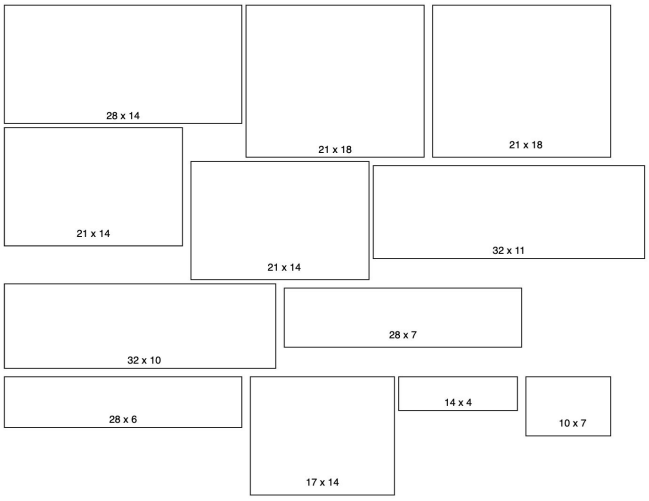
Requirements

- Must be written in Python 3.
- Must only use standard python libraries.
- Must include a README containing instructions indicating how to run the program and any assumptions made when developing the program.

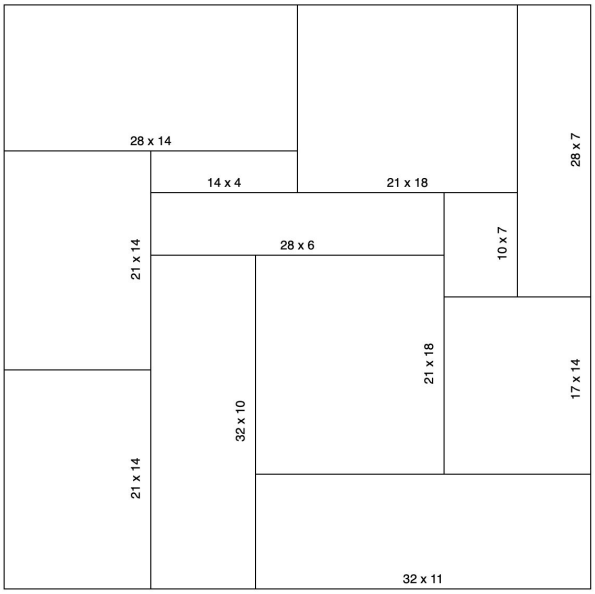
Example

Consider the following more complicated example. Suppose your program is given the following input which corresponds to the puzzle pieces on the right.

```
56,56
1,28,14
2,21,18
3,21,18
4,21,14
5,21,14
6,32,11
7,32,10
8,28,7
9,28,6
10,17,14
11,14,4
12,10,7
```



Your program should be able to determine that the pieces can fit perfectly into the 56 x 56 board and find one possible solution. One such solution is shown below. An acceptable output for your program indicating the position and orientation of each piece is displayed alongside.




```
~ python3 solve.py input.txt
Solution found.
Piece ID: 1, Position:(0, 0), Orientation: H
Piece ID: 2, Position:(28, 0), Orientation: H
Piece ID: 8, Position:(49, 0), Orientation: V
Piece ID: 4, Position:(0, 14), Orientation: V
Piece ID: 11, Position:(14, 14), Orientation: H
Piece ID: 9, Position:(14, 18), Orientation: H
Piece ID: 12, Position:(42, 18), Orientation: V
Piece ID: 7, Position:(14, 24), Orientation: V
Piece ID: 3, Position:(24, 24), Orientation: V
Piece ID: 10, Position:(42, 28), Orientation: V
Piece ID: 5, Position:(0, 35), Orientation: V
Piece ID: 6, Position:(24, 45), Orientation: H
```