I spent at least 80 minutes on Calibron but was unable to come up with a successful algorithm in order to solve the puzzles. Like I did for Rush Hour, I created a class in order to keep track of the variables. This class represented each block, and contained the attributes of the block’s width, height, and upper left coordinates that were represented by a X and Y. I also wrote up methods that would be able to change the position of the block, by changing where the current upper left coordinate of the block was. This would be done by simply changing the X and Y coordinates, if necessary. While this class contained much fewer methods than the Car class I created for Rush Hour, they were sufficient for what I was able to code. I was also able to code a way to determine if the blocks given in the command line were able to fit the actual board size. This was relatively easy compared to the other aspects of this lab. After reading all the blocks that were given through the command line, I was also able to make these blocks into a Block object, as well as create a rotated version of the block.

However, there were still many things that proved much more difficult to accomplish. For instance, while I had a general idea on how I could solve the puzzle, it was much harder to implement it in reality, as I also had a time crunch. My initial thoughts were that by keeping track of the upper left corner, width and height, I would also be able to determine where all the edges of an individual block were. I would be able to solve the puzzle by first placing a first block in the upper left, and then adding on blocks to either the right or bottom of the last placed block. While this was true, due to reasons that I am still not sure about, my list of Block objects would end up returning a list of tuples instead of a Block object. This made assigning locations to objects very difficult and oftentimes would not work for all cases. There was usually a ‘TypeError: 'NoneType' object is not subscriptable’ error that showed up in my assign method. My assign method was able to place a first block, but failed to place all other blocks. As most boards, besides the first test case, have multiple blocks, my assign method would not be sufficient in order to solve a puzzle. I think if I managed to circumvent this issue, perhaps through using a better data structure, I might be able to have a more effective assign method, and make further progress in this lab. In the future, if I ever have more time to work on past labs, I would also try working on creating a better algorithm in determining where to place blocks. I am pretty confident that my current algorithm has the potential to work correctly, but it does not seem very efficient, so I would like to try talking to another person who also wants to work on the assignment and see if we can come up with a better solution.