Aden Siebel and Adeline Yu

- describe the algorithm that you implemented, argue that it is correct, and argue its expected running time.
  Answer: In this algorithm, we find the most optimal solution for the sub-problems, from a sub-problem of 1 to the actual problem, finding the blocks that yield the greatest height and checking if it is stack-able. We store each solution in an array list, from the first sub-problem to then the last sub-problem (the entire problem). We return the last array, which contains the set of optimal blocks.
  Our algorithm is correct. We can show this by printing the results of the solution 0...(n-1), and then n. At base case, there are 0 blocks and 0 blocks to stack so we are done. If we assume that it is the most optimal solution at n - 1, then the optimal solution for n should include the solution to n - 1 and incorporate the next available block. We have sorted the blocks and we do a check for it being stack-able so we know it still holds the properties.
  Our algorithm is $O(n^2)$. Our algorithm will go through each element of the array and make comparisons with every other element.
- describe an interesting design decision that you made (i.e. an alternative that you considered for your algorithm and why you decided against it).
  Answer: We decided to sort the array of all possible block configurations by area. The area is the surface area of the block (length x width). This would be useful in reducing run-time and not performing some wasteful checks of whether a block is stack-able or not on another.
- an overview of how the code you submit implements the algorithm you describe
  Answer: The MakeTable() function in the BlockStacker class is how we mainly implement our algorithm. The Block class is mainly to set up the blocks (getters, setters, and a method to create additional configurations of the block). In the BlockStacker, we then have an array of possible choices of blocks and an array for our solutions at each number of blocks to include and consider. The MakeTable() function will iterate through each block in the array of choices in a nested for loop and add it to our solution and check if it is optimal compared to previous ones.
- how you tested your code and the results of sample tests
  Answer: We tested the inputs for 10 and 100 blocks and compared it with the expected outputs. When our code was yielding a different solution, we checked it by adding a final height tracker to both the outputs as well to see how off we were.