Olaolu Emmanuel
RUID: 159-00-3321
CS 111 Midterm Project Extra Credit

1. Average number of rounds needed to satisfy a board:(maxRounds set to 100)
   - Original Algorithm – 79 rounds
   - Improved Algorithm – 43 rounds

2. Values that are “better” for my improved algorithm:
   - My improved algorithm is designed to have a cell look for a spot where it could be satisfied as opposed to just looking for any blank spot like my original algorithm. However, if the threshold is too high and the cells can never be satisfied, then both algorithms will run very similarly because my improved algorithm will resort to using my old algorithm's method (moving to a random blank spot) if is not be able to find a spot where the cell can be satisfied. It is for this reason that my improved algorithm works better the more the threshold is lowered. Below, I have data showing a reduction in average number of rounds when the threshold is lowered.

   When threshold = 30: original algorithm = 79 rounds, better algorithm = 43 rounds (better algorithm is 45% better)
   When threshold = 10: original algorithm = 14 rounds, better algorithm = 5 rounds (better algorithm is 64% better)

3. Why I believe my better algorithm is better:
   - I believe my better algorithm is optimized because it will look to see if the unsatisfied cell can move to a blank spot where it can be satisfied as opposed to

just moving to a random blank spot which is what my original algorithm would do.

4. Comparison of efficiency:

   - Both algorithms run similarly except for the invocation of two extra methods in my improved algorithm. It invokes a method which checks each blank spot to see if an X cell would be satisfied if it moved there and if so, it assigns true to a 2D boolean array whose indices correspond to that of the tissue array. It invokes another method which does the same except for O cells. The worst case for both methods is 8n^2 (assuming that the entire tissue is blank spaces and it must check the eight spaces surrounding that blank cell). This results in a big-Oh of n^2. However, a method which is invoked in both algorithm's (isSatisfied), also has a big-Oh of n^2 so my better algorithm would have an efficiency of 3n^2 as opposed to just n^2, but in terms of big-Oh, constants can be removed so the big-Oh is the same. The average number of rounds for my better algorithm is almost half the amount of my original algorithm so I believe that the trade off is worth it.

5. What was my testing methodology:

   - To calculate #1, I used the constraints given along with setting the size of the board to a 10x10. First, I created a copy of the tissue so that each algorithm could use the same tissue sample ensuring that the only independent variables in the test were my algorithms. I ran the simulation 100 times by encasing my entire main method inside a for loop which would run 100 times. Each algorithm was encased in an infinite while loop which would be broken out of when either the max rounds (which I set to 100) was met or the board was satisfied. Before breaking from the loop, I add the number of round to a variable which would hold the total number of rounds for each algorithm (roundsNeeded1 and roundsNeeded2). On the last iteration of the simulation, I printed the total number of rounds needed for

each algorithm and compared the two.

6. Notes:

   - My CellSimImproved.java contains my original algorithm (moveAllUnsatisfied) and my improved one (moveAllUnsatisfied2). They were tested using the main method of that same java file.