My final project goals consisted of creating a 2048 game in python which I can use the expectimax algorithm to solve 2048. Firstly, I created a model of the 2048 game which i can use commands to move in a direction, then I implemented the game into pygame so I had a visual representation of the game. Creating the game took most of the time until now, but after I had verified that the game operates as it should I created an agent that just chooses random actions until the game is over. This will serve as a "control" to compare the other methods of solving the game. After that I implemented an agent using the expectimax function using the pseudocode from this source:
https://www.baeldung.com/cs/expectimax-search

I noticed that I was getting really low scores than what i originally expected (sub 10,000) so I tried tweaking the algorithm with heuristics. I implemented 4 heuristic functions, a snake heuristic (tiles follow a snake pattern- highest value tile should be at bottom left corner), a max tile in corner heuristic (rewards for having big values in the corners), an edge heuristic (rewards for aligning the tiles along an edge), and a empty cell heuristic (rewards more empty cells). Even with these heuristics the expectimax algorithm was not performing as well as I wanted to see, so I implemented Expectiminimax which does everything expectimax does but also calculates the expected values based on the probability of each tile being placed on an empty spot for nature's turn (chance nodes). Implementing this resulted in, on average, around a 3x in score.

Now that I have finished expectiminimax and expectimax (which were my goals for the project) I think I will implement deep Q learning to solve this game.

For the written part I have written an introduction and background on 2048, expectimax, and expectiminimax. I still have to write the methods, results and conclusion. I am planning on first implementing deep-q learning and then writing the rest of the paper.

Here is a list of things I still have to do (in order):

- Implement merge bonus heuristic
- Implement Deep Q Learning agent (time-dependent)
- Play a number of episodes with each agent (random, expectimax, etc) and generate score distribution charts
- Finish the paper