Chem 274B Final Project
Individual Reflection:

3.1. A description of their role in the project.

I coded the source files for our CA library and the field CA application implementation (I decided to go about coding the source file in not necessarily the most efficient way but in the most readable way to allow my group members to understand my logic and test it out themselves. I also edited the README and added titles to all our files.

3.2. How much do they contribute to the successful project completion (e.g., lead work, help others, coordinate meetings, etc)?

Since I coded most of the source files, I bore the responsibility of delegating what needed to be tested and how we could implement it for our application.

3.3. Describe any challenges/problems you and/or your team dealt with and how you solved Them.

We initially jumped straight into coding our implementation of the CA (pest control) rather than implementing the general-purpose library. Once we got that correction during the check-in, we quickly worked to implement the GPL for all 36 cases. Since we already had the logic figured out during our CA implementation, we just had to make it more general.

Furthermore, we had another issue when implementing our general-purpose library. We were working in the context of 3 different states(healthy, infected, dead). This was due to our initial plan for initializing the grid, where all squares are assumed to be filled with crops, and we studied different modes of transmission. However, we had to pivot to studying how crop density affects the spread of pests instead, as this would allow us to quickly implement the GPL where only one boundary and one rule condition are used. Hence, we had to edit the representation of the state transitions so that 0 meant no crops and would continue to stay at 0 regardless of the time steps taken. Then, the initialize function will populate the grid with 1 (or the presence of crops) at a certain density probability. Finally, similar to Dr Drummond's forest fire simulation, the square in the center of the grid is updated to 2 to represent (infected or the presence of crops).

Note: since tests were created and confirmed by my partners. We saw that it all worked together over zoom. However, their devices didn't require the additional -std=c++17 parameter that I personally need in order to validate the working code on my own device. Hopefully, i remembered to add them to all. But I made sure the main functions are still compiling and running on my own device.

3.4. Comment on the algorithmic and performance analysis of your library (you can base this on the data structures and functions you included). You do not need to provide a detail of every function that you implemented but rather provide a high level description of the design considerations that make you choose the data structures and algorithmic implementations that you used in this project. In particular, the parts of the project that you lead.

Vectors of vectors to hold grid:
I decided to use a vector to represent the grid over an array as vectors have extensive API that would allow for the use of built-in functions and provide the most flexibility in our implementation. Furthermore, crucially, vectors are easily assigned, meaning we can easily save the next grid as the current grid and vice versa. In contrast, you cannot assign an array to another array. Even though references and pointers are allowed, from my memory of class, when we were testing the same array of stored random integers, you asked us to split the array into two and copy the front half to the back, etc. I wanted to avoid the unnecessary reiteration completely and focused on allowing the vectors to be assigned to each other without the risk of memory leaks or deallocating pointers ever becoming an issue. Vectors also handle their own memory allocation and deallocation.
As a more seasoned Python programmer, I can easily understand and implement vectors.

Even though arrays are generally faster containers (stack allocation vs heap for vectors), I figured the overhead for a 20x20 grid (used in our implementation) wouldn't matter too much.

Furthermore, using integers to represent states instead of enum classes or strings to represent states inside the grid allows for more algorithmic efficiency memory and time-wise.
Strings are more costly to hold memory-wise and more complex to handle algorithmically than integers. Enum classes are also generally difficult to print to the console, which we need to use to check the functionality while building the code.

For loops
As I mentioned before, I coded everything out so that the logic was straightforward to follow for my group members. I wanted to ensure we could independently understand the logic and debug if necessary. However, loops are often costly in efficiency. Still, splitting the logic into discrete chunks depending on the condition type would allow it to bypass the non-pertinent loops and access the one that matches its condition.

Class structure:
We decided to employ a main Cellular Automata parent class and two derived classes, 1D_CA and 2D_CA, allowing us to create the same functions but in the scope of the current dimensionality. I decided to create functions only in the base CA class to avoid repeating similar functions. However, I realized that the additional dimension that 2d introduces (another wrapping for loop for the y-axis) would be highly unnecessary for the 1d case (in terms of temporal and memory efficiency). Hence, we decided to implement the more complex functions inside the derived classes individually. Both children classes should have the same functions

that perform the same functionality. This base and derived class structure allow us to introduce more derived/children classes that can account for more dimensions (3d, 4d, etc.).

3.5. Comment on how effective your showcase CA application from Problem 2 worked (i.e., how realistic was the model, any suggestions on how to improve its accuracy, etc.)

For our CA Application, we created a model that mimics the spread of pests in crops. We are hopeful to simulate how the initial density of crops affects the spread of pests. Our model uses the 2D version of our CA general-purpose library, with cutoff boundaries, as this is the most realistic way to model a field. Our crops have three potential states: empty, healthy, infected, and dead. We hope a model like this can help farmers find the optimal density to plant their crops to maximize yield and minimize the risk of spreading pests.

More information about our CA application can be found in the FinalProject_Assesement.pdf in the main directory.

It was effective. We decided that managing the crop density and assessing the risk of pest spread is a very important decision that farmers must make. On one hand, having the most bountiful harvest is their goal, yet if the crops are bad, it will all be moot. Our model showcases that even 5% differences in initial crop density can have very significant implications on the health vs infected vs death counts of the field after 40 days (see graphs). This highlights the delicate balance needed to maintain crop health whilst reducing pest spread risks.

It was realistic in that we used Von Neumann neighborhood, where infestation spread can only be done if the crops actually touch each other and an update_probability rule that can be adjusted to showcase the probability of infestation spread.

I believe that accuracy can be achieved by fine-tuning the model to suit the current type of field (may not be of a perfect square), the type of crops grown (may not even be all the same), the type of pests (can be more than one pest), the type of transmission (used soil/touching, can add air - with Von Neumann neighborhood) and finally crop resistance.

3.6. What you would have done differently :

3.6.1. Library Development

- I would have liked to add enum classes for each condition (boundary, neighborhood, rule) for our field implementation; however, we decided that sticking with integer representations would result in the easiest implementation of our general-purpose library. Enums would make the code much more user-friendly so the reader can clearly understand what conditions are being implemented.
- Maybe we can split up the larger functions in the GPL source file so that each function handles different conditions rather than all at once (ie step, find_neighbor functions)
- I wish we also had more time to add custom errors and follow best software engineering practices so that each function returns an error code (0 or 1)

3.6.2. Software project management

- I definitely wished I was better at collaborating on Github as I often was unsure of when to merge and how to resolve merge conflicts. However, Sean was awesome and hand-holding me through scary merge conflicts that might have deleted all my work.
- I also wished I left more comments in GitHub itself rather than scattered throughout the code, which would have allowed my team members to directly pinpoint the issues rather than checking the code and discord messages.
- I wished we had all read the instructions more carefully ahead of time so that we could have better-distributed work and knew exactly what files we needed, as the limited time we had post-check-in definitely led to some messy organization.

3.6.3. Product improvements

- It might be useful in the expansion of this model to create a struct that can hold specific information about each square in the grid.
- The struct could include all the information I mentioned above (in regards to improving the accuracy of our field model) as well as a health gauge that could track the days that it has been infected and its starting health (which indicates in part what type of crop it is and its innate resistance levels to pests). This, however, was beyond the scope of the project.
- I also hope that the struct would allow us to count for multiple forms of transmission, which would be much more necessary for a realistic implementation. The health gauge can also determine when the plant will die rather than using just a time-based step for the infection to death.

Overall, I learned SO MUCH in this class. It was extremely difficult at times, but our final project final product is a testament to the progress I have made as a software engineer. Ty to Dr Drummond, Manish, and Lewis for an amazing semester!