Project 1: Outbreak

Submission Due: October 24 by 11:59PM

Program Description:

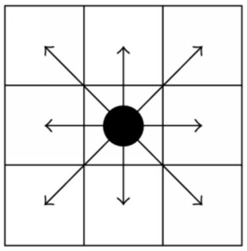
For this project you will be designing and implementing a system to simulate an outbreak across a geographic region. Specifically, you will be using a cellular automata, with a Moore neighborhood configuration, to model a geographic region, and a SIR model to model the health states of the populace. Your model should visually output how the region's population changes over time, the day of the outbreak's peak, and the day the outbreak ended.

In disease modelling, one of the basic models is called the SIR model. SIR stands for Susceptible, Infectious, and Recovered, which represent the different health states a person can be in with regard to an illness. A susceptible person is someone who is not sick, but can become sick. An infectious person is someone who is sick and can infect others with the disease. A recovered person is someone who is no longer sick and has developed immunity to the disease. For this model, we will be assuming that the immunity is permanent.

To simulate the disease progression across the geographic region, we will be using a cellular automata. Each cell represents one person's health state. Below is an example of a region with 25 people, 24 of which are susceptible (S), and one of which is infectious (I).

S	S	S	S	S
S	S	S	S	S
S	S	I	S	S
S	S	S	S	S
S	S	S	S	S

The above example represents the initial state of the region (i.e. Day 0) before the disease has begun to spread. In order to simulate the progression of the disease we need rules to determine which susceptible people (S) become infectious (I) on the next day. For this project, we will be determining a person's health state (S, I, or R) for the next day, based on their current health state and the current health states of the people around them. Specifically, a person's health state is determined by all people who are immediately adjacent to them, as can be seen in the following diagram. For this project, a susceptible (S) person will become infectious (I) on the next day, if a required number of people that are adjacent to the susceptible person are infectious (I) on the current day. Additionally, we assume the region is a flat plane, so people in the corners will only have three adjacent people, people on the borders will have five adjacent people, and people in the middle will have eight adjacent people.



Program Requirements

- Your program will require at least these two .py files, though you can create additional .py files if you find it necessary
  - **SimEngine.py**
  - **SimLibrary.py**
- Your **SimLibrary.py** file should contain:
  - A function to read in and store the configuration file's data, which takes in a `List` representing the region and
    - Prompts the user for the name of the file containing the initial simulation information, and if user enters a file that does not exist, repeatedly prompt for a new file name until the user provides one that does exist
    - Opens the file and tries to read in the data one line at a time and storing the data in appropriate data structures and variables
      - The first line indicates the required number of infectious people in a susceptible person's neighborhood for the susceptible person to become infectious on the next day, i.e. the threshold for infection
      - The second line indicates how many days a person remains infectious before they becomes recovered on the following day, i.e. if the infectious period is 2, then the person becomes recovered after being infectious for 2 days
      - All subsequent lines will be comma delimited lists indicating the starting health states of the different people in the region
        - s: susceptible
        - i: infectious
        - r: recovered
        - v: vaccinated
      - While the region will always be a rectangle, it could be a rectangle of any size
      - Be sure that you have a strategy for storing at least the health state and infectious period for each person that does not involve classes you've created
      - If any of the health states in the region are something other than s, i, r, or v, generate an appropriate exception with a message explaining that the program detected an invalid health state in the file
    - Catches any exceptions that may have been generated
      - If an exception was generated regarding an invalid health state, output the exception's message to the user and exit the program with a -1 error code
      - If any other exception was generated, output a message to the user informing them something was wrong with the input file and exit the program with a -2 error code
    - Once the file has been read in and the data stored, then, using an appropriate function, the initial state of the region should be output as Day 0
    - Returns the threshold data

  - A function to simulate the outbreak within the region, which takes in a `List` containing the region, the threshold for infection, and three `Lists` containing the daily counts of susceptible, infectious, and recovered people
    - While there is at least one infectious person, each day:
      - The simulation should check each person in the region to see if their health state needs to be updated
        - Vaccinated people never change health states
        - A person changes from susceptible to infectious if a sufficient number of people around them are infectious
        - A person changes from infectious to recovered if they have been infectious for the entire duration of their infectious period
        - Remember, a person's health state does not actually change until the next day
      - The entire state of the region should be output along with the day number using an appropriate function
      - The counts of each health state (s, i, r) should be added to the appropriate `List` (including Day 0 at the beginning of the outbreak)
    - After the simulation finishes, output with appropriate messages:
      - The final state of the region
      - The number of days the outbreak lasted (i.e. the number of days that had at least one infectious individual)
      - The highest count of infectious individuals and the day that occurred on. If multiple days have the same highest count, use the first day with that count
    - Return the number of days the outbreak lasted
  - A function to output the entire state of the region and the day number, which takes in a `List` containing the region and the current day number
    - The region and day number should be displayed in an easy-to-read format
  - A function to plot the results of the simulation, which takes in three `Lists` containing the daily counts of susceptible, infectious, and recovered people, and the number of days the outbreak lasted
    - Generate a single line plot that contains the counts of susceptible, infectious, and recovered individuals from day 0 until the end of the simulation, with one line per health state
    - Be sure to add an appropriate title, axis labels, and legend to your plot
  - Any additional functions you deem necessary and useful
- Your **SimEngine.py** file should:
  - Import your SimLibrary module
  - Contain a `main()` function which will:
    - Declare one, empty `List` to store the region
    - Declare three, empty `Lists`, to store the daily counts of susceptible, infectious, and recovered individuals

        - Call the function to read in and store the configuration file, passing in the appropriate information, and storing the returned threshold
        - Call the function to execute the simulation, passing in the appropriate information and storing the returned outbreak duration
        - Call the function to generate the plot of the outbreak, passing in the appropriate information
    - Call your `main()` function
- Your program should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, date, and brief description), comments for each variable, docstrings for each function, and commented blocks of code.
- You are not allowed to use any globally scoped variables in this project. All variables must be locally scoped.
- See sample input and output files to test and calibrate your program

Submission

- Your program will be graded largely upon whether it works correctly
- Your program will also be graded based upon your program style. This means that you should use comments (as directed), meaningful variable names, and a consistent indentation style
- You will submit all of your .py files to Canvas by the due date and time
- Projects will be accepted up to two days late, at a 10% penalty per 24-hour period after the due date
- Projects are meant to be individual coding assignments, so no collaboration is allowed. This includes downloading code off of the internet. Any discovered instances of this will be considered cheating and appropriate actions will the taken according to the course syllabus
- You are not allowed to use generative AI on this project. Any discovered instances of this will be considered cheating and appropriate actions will the taken according to the course syllabus
- Be sure that you have tested the version of the program you wish to submit to make sure it works correctly. You will not be allowed to resubmit work after the deadline that does not have a third-party timestamp such as Dropbox or an email. Timestamps generated by your operating system will not be acceptable.

Rubric

The entire assignment is worth 100 points and partial credit is possible. No credit will be given for portions of the program that cannot be tested due to the program crashing.

- ☐ **Program Executes Successfully**
  - If your program crashes due to syntax errors, you will lose 10 points, but we will attempt to fix minor issues (incorrect indentations, stray character, missing import) so that we can execute and test the program. We will not fix major issues that would require functionality to be further implemented, or a reorganization of logic in your code.
- ☐ **Simulation setup (25 points)**
  - Program prompts for, checks to make sure the file specified by the user exists, and repeatedly reprompts for a new file if it does not exist (5 points)
  - Program reads in and stores all of the input file data and outputs the initial state of the region and day number (10 points)
  - Program generates and handles appropriate exceptions (10 points)
- ☐ **Region update (25 points)**
  - Program correctly updates each person in the region, as necessary, each day (25 points)
- ☐ **Region output (10 points)**
  - Program outputs the state of the region and day number each day in an easy-to-read format (10 points)
- ☐ **Simulation summary (10 points)**
  - Program outputs the correct final day for the simulation and the correct total number of days the outbreak lasted (5 points)
  - Program outputs the correct peak day of infection for the simulation, and count of infectious people during that day (5 points)
- ☐ **Outbreak plot (10 points)**
  - Program outputs a line plot containing one line each for the count of susceptible, infectious, and recovered people across the days of the simulation
- ☐ **Instruction requirements met (10 points)**
  - The program contains at least the variables, functions, and files specified in the instructions (10 points)
- ☐ **Program contains sufficient comments across all files (10 points)**

Bonus

For 10 bonus points, write and submit an additional program named **inputFileGenerator.py** that will prompt the user for :

- The dimensions of the region
- The total population
- The number of infectious people
- The number of vaccinated people
- The threshold for infection
- The infectious period
- The name of the input file

It will then generate an input file using the specified name and information that conforms to the existing input files. Note, there should never be any recovered people in the initial region configuration.