Project 2: Disease Modelling

Milestone Due: October 24 by Midnight

Complete Project Due: November 7 by Midnight

Instructions

For this project you will be designing and implementing a system, in either C or C++, to simulate an outbreak across a geographic area. 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 final S, I, R, V counts, the day of the outbreak's peak, and the day the outbreak ended.

Additionally, sample input files will not be uploaded to Canvas, but can instead be found on the CSE machines at `/home/jeh0289/public/csce2100/fa18/proj2` You can `cd` into that directory and copy the input files from there.

For the project you may, if you choose, work in small groups of up to three students. You will need to sign-up for a group on Canvas by no later than midnight (10/10). This can be accessed in the Groups tab in the People section. Any student who does not join a group will be assumed to be working on their own and will be assigned a single member group. A group cannot be changed once created, unless it is completely dissolved. All members of a group will receive the same grade, and all members are expected to contribute equally to the programming and report. Only one submission per group will be required, but a group will not be penalized for having multiple members submit the completed project.

Also, as a reminder, all of the code for this assignment must be written by your group. You may not share code or download solutions off the internet, as doing so will be considered cheating.

Requirements

This assignment has two parts: a design portion and an implementation portion.

Design Document

For the design portion, you must generate documentation, in PDF format, describing your system and design process. The purpose of this is for you to explain not just what your system is doing, and how it is doing it, but why. You will need to justify your design decisions in a concise, informative manner. Justifications such as "I did this because it was easy" are not sufficient, as you should actually explain why a particular data structure or algorithm was more efficient, effective, or optimal. Additionally, commented code, while sometimes helpful in small examples, is not a sufficient explanation in and of itself. Your explanations and justifications are expected to be presented in prose and in paragraph format, i.e. not bulleted lists. Further, part of the evaluation of your design document is the apparent amount of thought and effort that went into its creation.

This document should be divided into four main parts, each with an appropriate header.

In the first part, you should describe your design process. Did you work out the algorithm on paper or a whiteboard before hand? Did you draw UML diagrams of the system? Did you create a small prototype? Did you simply start coding away and then recode once or twice with newfound understanding? In a few paragraphs, describe in detail how you went about designing the system, and be sure to provide sufficient justification of your methodology.

For the second part, you should describe the data structures you used in your system. What, if any, objects or structs did you create to store data? How did you organize and manage them? What types of formal data structures did you make use of (trees, graphs, arrays, hashes, etc)? In a few paragraphs, describe in detail how you stored the various data elements in your system, and be sure to provide sufficient justification of your methodology.

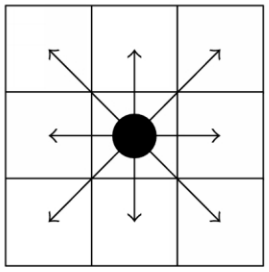
For the third part, you should describe functionality of your system. How is data moved and transformed? How is it read in? How is it output? What are the various major functions you constructed and how do they work? In a few paragraphs, describe in detail how your system works, and be sure to provide sufficient justification of your methodology. You might also consider including diagrams to more easily visualize how all of the pieces fit together.

Finally, you will need to briefly describe what share of the work was performed by each group member. Be sure to be specific about tasks.

Implementation

Your program must provide the following functionality and adhere to the following constraints:

- Allow the user to choose the file describing the initial setup of the simulation
    - The first line indicates the required number of infectious agents in a susceptible agent's neighborhood for the susceptible agent to become infectious on the next day
    - The second line indicates how many days an agent remains infectious before it becomes recovered on the following day, i.e. if the infectious period is 2, then the agent becomes recovered after being infectious for 2 days
    - The third line indicates the display frequently for how frequently the region should be displayed in days
    - All subsequent lines will indicate the starting health states of the different agents in the region
        - s: susceptible
        - i: infectious
        - r: recovered
        - v: vaccinated
- Agents that are one square away from some other agent are considered to be in that agent's neighborhood



- Agents on the left and right boundaries of the region should be considered adjacent, i.e. the region is not a flat map but more like a cylinder
- An agent can have one of four health states: susceptible, infectious, recovered, vaccinated
    - An agent can only move through the states as follows: S -> I -> R
    - A susceptible agent becomes infectious if they have threshold or greater number of infectious agents in their neighborhood on any given day
    - An infectious agent becomes recovered after infectious period number of days
    - Vaccinated agents are permanently vaccinated and cannot change states

- The simulation should run until the number of infectious agents is 0, indicating the outbreak has ended. There should only be S, R, and/or V agents left over
- The initial setup of the region represents the simulation at day 0
- Your program should output the following:
  - The initial status of the region, i.e. day 0
  - The status of the region every X days, where X is the display frequency from the input file
  - The final state of the region once the outbreak has ended, i.e. the first day that there are 0 infectious agents
  - The final counts of the number of susceptible, infectious, recovered, and vaccinated agents once the outbreak has ended, i.e. the first day that there are 0 infectious agents
  - The day the outbreak ended
  - The first day of the outbreak's peak in terms of number of infectious agents, i.e. the first day with the highest count of infectious agents
- Your code must be well commented.
- You must provide a short README file which includes your name and explains how to compile and run your program.
- Additionally, you may write a makefile if you want your code to compile with additional flags.

Suggestions

This project requires the use of a few different data structures to store and represent the agents within the model. First, you will need a way to store the grid of the cellular automata. This is a good opportunity to use either vectors or dynamic arrays, as you do not know how big the region is until you have read the entire input file. Then within that grid, you will need to represent each agent. Because each agent maintains both their health state (S,I,R,V) and how long they have been infectious, you should consider using a struct or object. As for the health states themselves, perhaps consider an enumerated type to make checking for states a bit more readable in your code.

Additionally, updating the grid from one day to the next could be done with two grids, one for the current day and one for the next day. Trying to update in place may cause problems, as an agent should not be considered infectious until the day after they have enough infectious agents in their neighborhood. Thus, consider using a temporary grid to determine what the state of the region will be the next day given the state of the region on the current day, and then once it is filled in, overwrite the current day with the next day.

Example Output

Threshold:1
Infectious Period:2
Display:1

Day 0

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

Day 1

s	s	s	s	s
s	i	i	i	s
s	i	i	i	s
s	I	i	i	s
s	s	s	s	s

Day 2

i	i	i	i	i
i	i	i	i	i
i	i	r	i	i
i	i	i	i	i
i	i	i	i	i

Day 3

i	i	i	i	i
i	r	r	r	i
i	r	r	r	i
i	r	r	r	i
i	i	i	i	i

Day 4

r	r	r	r	r
r	r	r	r	r
r	r	r	r	r
r	r	r	r	r
r	r	r	r	r

Milestone Submission

Your program must be able to output the region for day 0, and provide the counts of the initial number of S,I,R, and V individuals. You must submit a .zip file containing the following:

1. All files necessary to compile and run your program
2. A README file explaining how to compile and run your program

Complete Project Submission

Your program must provide all requested functionality. You must submit a .zip file containing the following:

1. All files necessary to compile and run your program
2. A README file explaining how to compile and run your program
3. Your design document in PDF format

Rubric

The entire assignment is worth 100 points. The breakdown of those points is as follows.

- 50 points: Design documentation
- 40 points: Code satisfies requirements
- 10 points: Professional coding style
  - 5 points: Adequate comments
  - 3 points: Modularity
  - 2 points: Readability
- If your code fails to compile on the CSE machines you may not receive credit for the programming portion of the assignment. I recommend not making changes to your code without checking for compilation before you submit.

Bonus

For 10 bonus points, your system should be configured to allow for the user to run the model as either a SIR or a SIRS model. In a SIRS model, agents eventually lose their immunity and become susceptible after some time. Vaccinated agents always have immunity though. In this case, your system should provide users the option of selecting which mode they want, SIR or SIRS, and then read in two additional lines in the input file. One will have the recovery period which determines how long an agent stays recovered before becoming susceptible. The other will have the maximum number of days the model will be allowed to run. Because allowing recovered agents to become susceptible potentially allows the disease to perpetuate in the system, the model should run until either the number of infectious agents is 0 or the maximum number of days has been reached.