Overview

I created the final disease simulation incrementally, starting with the person class.

This person class serves as the fundamental building block of the overall simulation, details of which are available at the bottom of this section.

Main 1 (main1.cc):

This first implementation of infection used only the person class. This function instantiated one person, “Joe”, who is guaranteed to be infected on the first day. The program then propagates Joe’s state forward until he is recovered, which will take 5 days. This script served as a test bed to demonstrate the functionality of the person class’ functions as well as my implementation of probability (generating a random number, then dividing by the maximum number the random generator could produce.)

Main 2 (main2.cc):

Once I demonstrated that the person class was functional, I created a population class, who’s final form is expounded upon at the bottom of this section. This class primarily served as a container for a vector of persons and the methods which would act on it. I tested this class recreating the one-person infection, which was printed to the screen using the population class’ method print_pop.

Main 3 (main3.cc):

disclaimer: this program preforms identically to main 4 due to updates made to dependent functions within the population class.

Main 3 introduced the spread of disease to direct neighbors. Through a similar loop to the above two programs with the addition of an input threshold probability, which, when activated by a randomly generated number, would infect a person who was directly neighboring an infected individual.

Main 4 (main4.cc):

This program expanded the capabilities of main 3 in two phases. The first was inoculation, which immediately rendered the method of spreading in the above program moot as one vaccinated individual completely stopped the spread of the disease. However, even without vaccination, a low probability of propagation would allow some persons to avoid infection before the illness has run its course. This barrier was overcome once the disease vould be propagated to both direct neighbors as well as to up to 6 random members of the population. In main 4, the population class had private members count and prob, which represented the population size and probability of propagation respectively. A graph in the following section reflects the herd immunity as demonstrated by main 4, this program represents a coherent submission that meets the base requirements of the assignment. However, my simulation is further refined and optimized in the following section.

Main T (maint.cc):

In this main program, I replaced the staging mechanism in main 4, which consisted of a candidacy vector and infection queue vector, with additional values stored within the person class. I had hoped to solve a prevalent but persistent issue with core dump when the probability of infection and percent of population vaccinated were not near one another (0.9 & 0.1, for

instance). Many of these changes actually occurred within the population class, but I also updated my main program to provide data both to the terminal and write to a file, which became useful with the introduction of bigtest.sh, a bash script which automated the running of different combinations of of inputs for my program. While I have a large csv text file of these results, I was unfortunately unable to import the data successfully into matlab. The data is still included in my project directory should the grader be interested in viewing it.

Person Class

Status Integer meanings:

- -4: candidate for infection (in maint)
- -3: Queued for infection (will be infected at cycle end) (maint)
- -2: the individual is vaccinated
- -1: the person has recovered from the illness
- 0: the person is susceptible to infection
- N > 0: the person is infected, with N days remaining

This class also contains the methods:

- getstatus: returns the person's status
- setstatus: sets user status. Used for vaccination and troubleshooting
- status_string: Prints a character representative of the persons status
    - X corresponds to vaccinanted
    - - represents recovered
    - ? represents succeptibility
    - + demonstrates the person is infected
- update: propagates the person forward (decreases infection time remaining)
- infect: infects the person, based on an input integer
- is_stable: returns true if status is < 1

Population Class

- set_probabilliity: given a probabililty 0-1, this sets the likelihood of spread
- random_infection: infects one random, non-vaccinated person
- random_vaccine: vaccinates a percentage of the population based on input 0-1
- count_infected: counts how many population members are infected
- update_spread: propogates the infection and updates time remaining for infected
- update: only updates those infected; included in final frm for testing
- print_pop: prints the population using the person's status_string method
- Survivors: returns how many people were vaccinated, infected (recovered), and spared infection.

Results

(For % inoculation = probability of infection)

Population = 1000