1 Problem Statement

Suppose we wish to develop an in-depth simulation for how illness spreads throughout a population. The best starting place when building such a simulation from scratch is from small, toy examples, in which one can easily add additional functionality and complexity to. Below, I have outlined a sequence of tasks and milestones for how one may go about creating such simulations.

**Milestone 1: Get a minimal “Person” class set up and working**

- Create a class called “Person” which contains a constructor, a default constructor, as well as several private data members:
  - PersonID
  - x-location
  - y-location
  - Age
  - (bool) Sick

  The non-default constructor should accept these inputs, in the event we wish to specify the initial conditions for the start of the simulation

- Write a source code which instantiates 2 instances of the Person class, and prints out their location, their age, and whether or not they’re sick. Note that this will require additional functionality defined for the class, such as “get_age()” and “get_condition()” which return their age and health status.

**Milestone 2: Create functions which allow the people to move around**

- Now we wish to have these two people “move-around”. Let’s assume they live on a grid, and will move either ±1 in the x and y directions, at each iteration. To implement this, write a member function for the Person class which randomly adds or subtracts 1 from the Person’s x and y coordinates each time its called.
• Within the source code, write a for loop which calls on both people to move 10 times. Have their position print out after every step.

**Milestone 3: Restrict the people to a finite box, let illness be transferred**

• We don’t want our simulated people floating out infinitely far away, so impose constraints on the coordinates and move function which restrict the people to an \( N \times N \) grid.

• It’s only a matter of time before these people now come into contact. In the source code, define a contact variable such that if the people come within a certain distance (say, within 6 units from one another), AND if one of those people are sick, then have both people become sick. To check the people’s distance, you will need an additional function in the source code checking the distance between two (x,y) coordinates.

• Increase the number of iterations for which the people are moving, stop printing out their position, and instead print out a message when the health status of an individual changes.

**Milestone 4: Let the people out and see what happens**

• Time to scale up, two people isn’t interesting (though you could take statistics on how long it takes for the two people to come into contact, as a function of the contact parameter). Let us make a vector that holds Person objects. Fill that vector with \( M \) people (I’d start with \( M = 5 \), and then increase once I’m confident its working), and have those people start with random initial positions, and let the first person be the only sick person initially.

• Now for each iteration, move all of the people (you’ll need a nested for-loop), check whether or not all pairs of people are within the distance threshold required to transfer the illness. If so, adjust the health statuses as appropriate. Measure how many iterations it takes to have all Person’s become ill. How does this number change as more people get introduced into this simulation?

• You now have a simulation where \( M \) people are random walking around a finite \( N \times N \) grid, and if they come within distance \( d \) of each other, they fall ill. Just understanding how your choices of \( M \) and \( d \) change the dynamics of the health statuses (sometimes everyone gets sick quickly, sometimes it happens quite slowly), is an interesting thing to study.

If you’ve gotten to this point, get creative, what happens if people recover after every \( T \) iterations? What happens if some people have a mask, thereby decreasing the distance parameter? We could add randomness to whether or not people contract the illness after coming into contact, i.e. if a random number is above a threshold, then illness gets transferred. Transfer probabilities could be a function of age. The list of possibilities are endless.