PIC 10A Section 2 - Homework #8 (due Monday, February 28, by 11:59 pm)

You should upload each .cpp (and .h file) separately and submit them to CCLE before the due date/time! Your work will otherwise not be considered for grading. Do not submit a zipped up folder or any other type of file besides .h and .cpp.

Be sure you upload files with the precise name that you used in your editor environment otherwise there may be linker and other errors when your homeworks are compiled on a different machine. Also be sure your code compiles and runs on Visual Studio 2022.

"SIR MODEL: EFFECTS OF MORONS IN THE EARLY COVID ERA"

In the end you will submit **sir.h** which defines an **SIR** class and declares its various member functions/constructor and **sir.cpp** which provides the definitions of the various member functions/constructor of the **SIR** class.

This homework focuses on writing a class to simulate the spread of COVID-19 *before vaccines were available*. One **important remark:** the model we will work with is highly simplified. While our results may be qualitatively descriptive of real-world scenarios, the quantitative accuracy is highly uncertain. Real life is complicated and modelling involves making some simplifying assumptions.

We will investigate the spread of COVID-19 paying attention to three disease states: Susceptible (those who have not been infected yet), Infected (those currently infected and who are contagious), and Removed (those who have recovered or died). We also consider two types of people: normal people who wear masks and reduce their contacts and morons who do neither. We will retrospectively see how varying the percentage of morons in society affects the overall spread of the disease.

Note: you should consider what the instructions are saying. This is similar to the example from class notes but it deviates from that example in a number of ways, notably that the time steps are not only in chunks of one day: the time step interval Δt plays a role. Also, there are more compartments in society besides S and I. But it is also more similar than it may seem. In the notes, we simply used member variables **S** and **I** to denote the number of susceptibles and infected people at a given time (whatever time the class is storing). In the notation, you will see equations involving $s_n(t)$, $i_n(t)$, etc. The t does not really mean they are explicit functions of time. The t merely indicates they

change as time moves forward, just like the **SIS** class in the notes.

Some nomenclature is defined below, as are the rules for how fractions of the population in different disease states changes from one day to the next.

For those of you interested in the model derivation, some details are included at the end of the homework, after the assignment has been presented.

We denote:

- t: the elapsed time in days since the start of the epidemic;
- $s_n(t)$: the fraction of the population who are susceptible normal people at time t;
- $s_m(t)$: the fraction of the population who are susceptible morons at time t;
- $i_n(t)$: the fraction of the population who are infected normal people at time t;
- $i_m(t)$: the fraction of the population who are infected morons at time t;
- $r_n(t)$: the fraction of the population who are removed normal people at time t;
- $r_m(t)$: the fraction of the population who are removed morons at time t;
- c_n: the daily number of contacts for normal people, assumed constant;
- c_m: the daily number of contacts for morons, assumed constant;
- P: the total number of people in the city;
- f: the fraction of people in the city who are morons (so fP is the number of morons and $(1-f)P$ is the number of normal people);
- $N = (1-f)P$ the number of normal people;
- $M = fP$ the number of morons;
- i^*: the fraction of people who are infected at time $t = 0$;
- β (beta): the probability of getting COVID-19 when interacting with a sick person on a given day;
- μ (mu): the risk reduction factor **experienced by those interacting with someone wearing a mask**; and
- γ (gamma): the daily rate people recover/die from the ill state and move into the removed group.

For convenience, we denote (the greek letter used below is Psi):

$$\begin{aligned}\Psi &= Nc_n + Mc_m \\ \Psi_m &= Mc_m/\Psi.\end{aligned}$$

Given i^*, we choose initial values:

$$\begin{aligned}s_n(0) &= (1-i^*)(1-f) \\ i_n(0) &= i^*(1-f) \\ r_n(0) &= 0 \\ s_m(0) &= (1-i^*)f \\ i_m(0) &= i^*f \\ r_m(0) &= 0\end{aligned}$$

Given $s_n(t), s_m(t), i_n(t), i_m(t), r_n(t)$ and $r_m(t)$, their changes from time t to $t+\Delta t$ are given by:

$$\begin{aligned}\Delta s_n &= -H_n s_n(t) \\ \Delta i_n &= H_n s_n(t) - Ri_n \\ \Delta r_n &= Ri_n \\ \Delta s_m &= -H_m s_m(t) \\ \Delta i_m &= H_m s_m(t) - Ri_m \\ \Delta r_m &= Ri_m\end{aligned}$$

where

$$\begin{aligned}H_n &= 1 - \mathrm{e}^{-\beta \Delta t c_n \left(\frac{\mu(1-\Psi_m) i_n(t)}{1-f} + \frac{\Psi_m i_m(t)}{f}\right)} \\ H_m &= 1 - \mathrm{e}^{-\beta \Delta t c_m \left(\frac{\mu(1-\Psi_m) i_n(t)}{1-f} + \frac{\Psi_m i_m(t)}{f}\right)} \\ R &= 1 - \mathrm{e}^{-\gamma \Delta t}.\end{aligned}$$

By "their changes from time t to $t+\Delta t$," we mean that, for example

$$s_n(t+\Delta t) = s_n(t) + \Delta s_n$$

(so that in code, you may have something like `sn += delta_sn;`), etc., etc., etc.

Task: write an **SIR** class that manages the model statistics. The class must:

- have **double** values to represent the current time t along with the six fractions $s_n(t)$, ..., $r_m(t)$;
- have a **single constructor** taking in values for c_m, c_n, P, f, i^*, β, γ, and μ as described above and initialize $t = 0$ and the other fractions as described above;
- a function **step** accepting a value of Δt and advancing the simulations forward that length of time in days;
- a function **get_time** to return the current time in days;
- a function **get_sn** to return the current fraction of susceptible normal people;
- a function **get_in** to return the current fraction of infected normal people;
- a function **get_rn** to return the current fraction of recovered normal people;
- a function **get_sm** to return the current fraction of susceptible morons;
- a function **get_im** to return the current fraction of infected morons; and
- a function **get_rm** to return the current fraction of recovered morons.

Note: there are special cases that emerge when $f = 0$ and $f = 1$, but *do not worry about those cases here!* Your code will not be tested on those cases!

A main routine is given to you. Your code may be tested on a different main routine. See an example of the code running below.

Remark: the values depicted are based on loose estimates for COVID-19 when available. Here is an example main routine with the desired output. If your values do not match what is shown here exactly, your code is wrong. It is not an issue of roundoff error. You should match exactly. Also note that **dT** is not required to be equal to 1 when your code is tested!

```
#include<iostream>
#include ‘‘sir.h’’

int main() {

  const double cm = 14; // normal, without distancing
  const double cn = 9; // after distancing

  const int P = 1000000; // 1 million people, say
```

```
const double istar = 0.005; // 0.5% are initially infected
const double beta = 0.02; // /day
const double gamma = 1./14; // /day
const double mu = 0.34; // if wearing mask

const double f1 = 0.001; // 1 in 1000 are morons
const double f2 = 0.05; // 1 in 20 are morons

const double Tend = 300; // run till 300 days, say
const double dT = 1.; // time steps of 1 day

SIR p1(cm, cn, P, f1, istar, beta, gamma, mu);

while (p1.get_time() < Tend) { // until simulation end time
  p1.step(dT);
}

// calculate fractions of different populations no longer susceptible
const double frac_to_pct = 100;
const double normal_percent_infected_or_recovered1 = frac_to_pct *
  (1-f1 - p1.get_sn())/(1-f1);
const double moron_percent_infected_or_recovered1 = frac_to_pct *
  (f1-p1.get_sm())/f1;

std::cout << "With a fraction " << f1 << " being morons, by day " <<
  p1.get_time() << ", " << normal_percent_infected_or_recovered1 <<
  "% of normal people have been infected and "
  << moron_percent_infected_or_recovered1 << "% of morons have been infected.\n";

SIR p2(cm, cn, P, f2, istar, beta, gamma, mu);

while (p2.get_time() < Tend) { // until simulation end time
  p2.step(dT);
}

// calculate fractions of different populations no longer susceptible
const double normal_percent_infected_or_recovered2 = frac_to_pct *
  (1 - f2 - p2.get_sn())/(1-f2);
const double moron_percent_infected_or_recovered2 = frac_to_pct *
  (f2-p2.get_sm())/f2;

std::cout << "With a fraction " << f2 << " being morons, by day " <<
  p2.get_time() << ", " << normal_percent_infected_or_recovered2 <<
  "% of normal people have been infected and "
```

```
    << moron_percent_infected_or_recovered2 << "% of morons have been infected.\n";

  return 0;
}
```

```
With a fraction 0.001 being morons, by day 300, 3.7285% of normal people have been infected and 5.47662% of morons
been infected.
With a fraction 0.05 being morons, by day 300, 17.5083% of normal people have been infected and 25.6672% of morons
been infected.
```

Model Derivation (for those interested)

Some mathematical details are left out, but this should be enough to understand where the equations come from. We assume for simplicity that all normal people have the same number of daily contacts c_n and all morons have the same number of daily contacts c_m. Ignoring some details about network configurations, there are Nc_n connections to normal people in the network and Mc_m connections to morons (just multiply the population counts by the number of contacts each has). Thus, the probability a susceptible person chosen at random is interacting with a normal person is $\frac{Mc_m}{Nc_n+Mc_m} = \Psi_m$ (i.e. fraction of contacts that are normal) and similarly, the probability they are interacting with a moron is $\frac{Nc_n}{Nc_n+Mc_m} = 1 - \Psi_m$.

To model transmission, we aim to estimate the *hazard rate* (probability of getting the disease per unit time) of a susceptible person getting the illness. If the hazard rate is λ then over a time period Δt, the probability of getting COVID-19 is $1 - \exp(-\lambda \Delta t)$. Due to different types of interactions and the fact the number of infected people changes over time, λ will vary.

We assume that without a mask, when an ill person interacts with a susceptible person, they contribute a total β to that susceptible person's hazard rate, with a risk reduction factor μ if that ill person is wearing a mask. Thus, *on average*, we would expect the hazard rate for a normal person to be:

$$\begin{aligned}\lambda &= c_n \times (\beta \Pr(\text{interact with infected moron}) + \mu\beta \Pr(\text{interact with infected normal})) \\ &= c_n \times (\beta \overbrace{\Psi_m}^{\text{with moron}} \overbrace{\frac{i_m}{f}}^{\text{porotion of morons infected}} + \overbrace{\mu}^{\text{reduction}} \beta \overbrace{(1-\Psi_m)}^{\text{with normal}} \overbrace{\frac{i_n}{1-f}}^{\text{portion of normals infected}}).\end{aligned}$$

The same result holds for morons but with c_n replaced by c_m. We generalize the hazard to arbitrary time steps Δt, not necessarily 1 day. This is the model presented.

Model Limitations

As George Box once apparently wrote, "all models are wrong but some are useful." The model we worked with is wrong... but could still yield useful insights. A few things that are clearly not considered in the model but could be included:

- We overlooked the possibility of people who have been exposed but are not infectious and did not distinguish between those in the prodromal/asymptomatic stages and those who are symptomatic.
- We assumed the population is well-mixed, without regard for the actual contact structure in society.
- We assumed all masks are equally effective and those wearing masks wear them correctly.
- We allowed time to be incremented in arbitrary steps of Δt but $\Delta t = 1$ day is perhaps more realistic.
- And there is of course the question of what the model parameters such as β, γ, etc., should be — and the literature gives a wide range of possible values for all of these.

With this in mind, you can better reflect on what the model is telling us and how accurate some of the results are.