Overview

For this assignment, you will write your own 1-D Monte Carlo particle transport code. In doing so, you will exercise all of the C++ skills we've learned so far. The range of inputs and physics will be very simple so that we can constrain the problem and make something that actually works.

As you think about how to build this application, it should quickly become clear why this is such a complex problem! Building a code like this that is performant, extendable, and easy to use is quite a challenge.

Our particles will only be able to move in one direction along a 1-D track. There is only 1 interaction type for our particle: absorption, which is characterized by a per-unit-step absorption probability. That is, every step of the track, the particle has that probability to be absorbed.

Code Submission

Create a folder in your Github repository called **assignment3/**. Here, you should only have 4 items

1. A **src/** directory
2. An **include/** directory
3. CMakeLists.txt
4. An **images/** directory

This is all that should be needed to compile and run your code. Your .hpp and .cpp files will be in include/ and src/, respectively. In CMakeLists.txt, call your executable **mc1d** so it will be easy for me to run with a script. The **images/** directory will be for your histograms of the results. Anything else (build/ directory, etc) should be in your .gitignore.

This assignment will be due by **Tuesday, September 28th at midnight Pacific Time.**

Design

You may use the CMakeLists.txt and **ArgParser** class we created during lecture, however it will be a beneficial exercise to write it yourself:
https://github.com/micahfolsom/ne697-lectures/tree/main/lecture10/mc1d

You should have these other classes/structs in their own source (.hpp and .cpp) files: **RunManager**, **Particle**, and **Hit**.

RunManager: high-level interface, manages everything

- run()
- write_results()

Particle: (class) this is what we're transporting

- Particle id
- Current position
- Alive or dead status

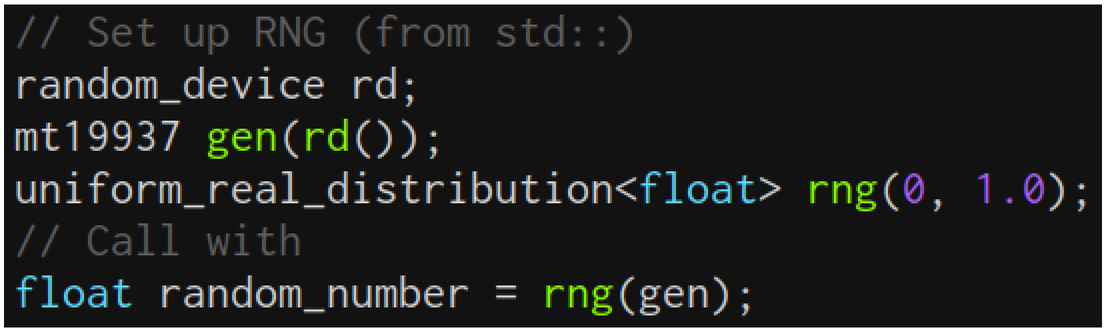
Hit: (struct) we will use these to store the hit positions along the track
- Position of absorption

Your program should take three command-line arguments provided in this order:
1. Track length
2. Absorption probability per unit length
3. Number of particles to run

You should boundary-check the inputs. You can assume they are numbers.
Track length will have a maximum value of 100.
Absorption must be > 0 and <= 1.
Number of particles must be >= 1.

RunManager::run() should contain your transport loop, where hits are recorded
RunManager::write_results() will tally everything, print a histogram, and write to disk

You can use the **<random>** header file to generate random numbers (there is a "using namespace std;" statement somewhere above this code):



```
// Set up RNG (from std::)
random_device rd;
mt19937 gen(rd());
uniform_real_distribution<float> rng(0, 1.0);
// Call with
float random_number = rng(gen);
```

Expected Output
Your program should produce helpful error messages (caught exceptions) when invalid numeric inputs are entered, or not enough/too many arguments are provided.

When the program starts, it should print out the operating parameters.

When it ends, it should print out a short summary of the results. An ASCII histogram like shown in class is not required (though you're encouraged to try).

Your program should write out a file, **hits.csv**, that contains a list of positions of particle absorptions (single column csv with no header). You can use whatever program you're comfortable with to make a histogram of the results.

Deliverables
Your code must be submitted to Github using the directory structure noted above. I will compile it and run it with a range of parameters, then confirm that the hits.csv outputs are statistically correct.

Run your program with the following commands:

```
./mc1d 50 0.8 100000
./mc1d 75 0.2 100000
./mc1d 90 0.05 100000
```

Make a histogram of the results for each, then save the images (.jpg, .png, etc) of your histograms to the **images/** directory and commit them to your repo. These should be e.g. **run1.png, run2.png, run3.png**. You do not need to save the hits.csv files that produced the histograms.