The goal of this assignment is to compute the many-particle dynamics of an interacting classical system. You should begin by downloading the Assignment2.tar.gz archive from the class website. In the Assignment2 directory there are two header files, vec2.hpp and particle.hpp, which contain class definitions for a two dimensional vector and a point particle. You are free to use these classes or not and to modify them in any way you like.

As an example of how to proceed, I have included a program solar.cpp, which simulates three small “planets” orbiting a larger “star” in an artificial (and unstable) solar system. All four bodies interact via the gravitational force (with $G = 1$) and are confined by their initial conditions to the $x$-$y$ plane. From the Assignment2 directory, the commands

```
$ make
$ ./solar
```

will build and execute the program. As you will see, solar.cpp contains OpenGL code that opens a new window and animates the planetary motion. The window displays everything inside the coordinate square with vertices (-1,-1) and (1,1). You can adjust the speed of the animation by recompiling with different values of `const int delay`.

1. (5 points) To start, modify solar.cpp to execute the choreographic three-body solution, described in Phys. Rev. Lett. 98, 201102 (2007). Instead of the example planetary system, set up three particles of unit mass with the following initial positions and velocities.

$$
\begin{align*}
  \mathbf{r}_1 &= (0.48500218, -0.121543765) & \mathbf{v}_1 &= -\frac{1}{2}\mathbf{v}_3 \\
  \mathbf{r}_2 &= -\mathbf{r}_1 & \mathbf{v}_2 &= -\frac{1}{2}\mathbf{v}_3 \\
  \mathbf{r}_3 &= (0, 0) & \mathbf{v}_3 &= (-1.31862315, -1.222914959)
\end{align*}
$$

(Hint: The collection of particles is stored in a global vector variable named gas. Particles are added in the function initialize_gas, using the push_back method.) Next, have the program dump the trajectory of one of the particles to a file. Plot the $x$ coordinate of the particle versus $t$ and extract the period. (Hint: The $\ll$ is overloaded in the particle class definition, so `cout $\ll$ gas[n]` is a legal way to output information about the $n$th particle.)

2. (10 points) We now want to investigate the behaviour of a two-dimensional gas of particles interacting via Van der Waals forces. The energy has the form

$$
E = \sum_i \frac{1}{2}m\mathbf{v}_i^2 + \sum_{i<j} U(r_{ij}),
$$

where the indices $i, j$ range over the particle number, $r_{ij}$ is the distance between two particles, and $U(r)$ is given by the famous Lennard-Jones potential:

$$
U(r) = 4\epsilon \left[ \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6 \right].
$$

The force between a pair of particles,

$$
F = -\nabla U(r) = \frac{24\epsilon}{r} \left[ 2 \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6 \right],
$$
vanishes at \( r_0 = 2^{1/6} \sigma \). Particles closer than \( r_0 \) repel each other and those farther apart attract. Implement this new force law in your code using the values \( \epsilon = 1 \), \( \sigma = 0.21 \).

Unlike the solar system, the gas will need to be contained. Restrict the particles to a \( 2 \times 2 \) box by implementing periodic boundary conditions for the coordinates \(-1 < x < 1\) and \(-1 < y < 1\). You will have to modify the concept of distance accordingly.

Next, initialize the gas by arranging 100 particles at rest in a \( 10 \times 10 \) square grid with spacing \( \Delta x = \Delta y = 0.2 \):

\[
\begin{bmatrix}
(-0.9,0.9) & (-0.7,0.9) & (-0.5,0.9) & \cdots & (0.9,0.9) \\
(-0.9,0.7) & (-0.7,0.7) & (-0.5,0.7) & (0.9,0.7) \\
& & & \vdots & \\
(-0.9,-0.9) & (-0.7,-0.9) & (-0.5,-0.9) & \cdots & (0.9,-0.9)
\end{bmatrix}
\]

Introduce small random deviations (of order 0.01) in the positions. You should use the provided random number generator

```
#include "mtrand.hpp"
mtrand Rand;
double R(void) { return 2.0*Rand() - 1.0; }
```

to place particles at \( \text{vec2}(x+0.01*R(),y+0.01*R()) \).

With these initial conditions, the system is actually a solid rather than a gas. The square lattice pattern is merely meta-stable; it is not the ground state. The slight random deviations you added allow the atoms to rearrange into a new crystalline pattern. What is it?

3. (5 points) The animation routine puts the program in an endless loop. Introduce a global counter than keeps track of the total elapsed time \( t \). Have the program exit when \( t > 20 \). Use the \texttt{exit(0)} function call in the C standard library:

```
#include <cstdlib>
using std::exit;
```

Generate a histogram of all the pair distances in the time window \( 10 < t < 20 \). This quantity is called the radial distribution function. Plot it and discuss its features.

4. (5 points) Initialize the particles in their \( 10 \times 10 \) grid as before, but reduce the atomic “size” to \( \sigma = 0.1 \) (in effect creating a more dilute gas) and add small deviations (of order 4) to the initial particle velocity, e.g., \( \text{vec2}(4.0*\text{R}(),4.0*\text{R}()) \). Measure a velocity histogram in each of the time intervals \( 1 < t < 2 \), \( 4 < t < 6 \), \( 9 < t < 12 \), and \( 16 < t < 20 \). Compare these with the two-dimensional Maxwell-Boltzmann distribution,

\[
f(v) = \frac{mv}{kT} \exp\left(-\frac{mv^2}{2kT}\right).
\]

Estimate the time scale for the system to thermalize. Can you perform a fit to extract the temperature \( kT \)?

5. (5 points) What does the radial distribution function look like for the dilute gas? Plot it alongside the analytical expression for a uniform, non-interacting gas. What do the deviations reveal?