1. **Diffusion versus dispersion errors.** Implement the Lax-Friedrichs, Lax-Wendroff, and first order upwind methods for solving the scalar advection equation,

$$u_t + a u_x = 0 \tag{1}$$

in one dimension. Evolve a discontinuous initial condition ($u_0 = 1$ for $|x| < 1$, $u_0 = 0$ otherwise) on the domian $-2 < x < 2$ using periodic boundary conditions and 100 grid points to time 20 using $a = 1$. Plot the resulting solutions for a CFL number of 1.0, 0.8, and 0.5. Discuss.

2. **Linear Wave Convergence.** Download and install the Athena3.1 MHD code. There is a User's Manual for the code in the `/doc` directory which you can read to learn the basics of how to use Athena. Test your installation by running `configure` and then `make all` and `make test`; the code should run successfully and print a small number.

Now trying running a linear wave convergence test. Configure the code with `configure --with-order=3 --with-flux=roe --with-problem=linear_wave1d` and then compile with `make all`. (you should always use `make clean` before configuring and compiling the code for new problems). This will generate an executable in the directory `/bin`. The input file for 1D MHD linear wave convergence problems is `/tst/1D-mhd/athinput.linear_wave1d`. There is a script in the same directory which can be used to run multiple jobs at different resolutions. Edit the script to run only "wave" 0 (the left-going fast magnetosonic wave) for resolutions up to 1024. Note that with Athena any parameter value in the input file can be changed at run time using the command line, and this script exploits this feature. The L1 error norm is printed in a file in the directory `tmpdir.LinWave` when the script is run, located in the same directory as the script. Plot the error versus resolution and measure the convergence rate.

If you have time, try playing with other parameters (e.g. use "wave" 6, the right-going fast magnetosonic wave, which should be identical to the above results). Also try configuring the code with `order=2`, and/or `flux=hlld` to see how different orders of reconstruction or different Riemann solvers affect the error.

The problem generator for this test is in `/src/prob/linear_wave1d.c`, feel free to browse the source code to see how this problem is set up.

3. **Wave Steepening.** It is easiest to study the formation of shocks in hydrodynamics. Reconfigure the code with `configure --with-order=3 --with-flux=roe --with-gas=hydro --with-problem=linear_wave1d`, and then compile the code. Using the input file in `/tst/1D-hydro/linear_wave1d`, edit the wave amplitude in the `<problem>` input block to be 0.1. Run the problem, and use the .tab files to plot the profile of the density. How long does it take for the initially smooth sinusoidal density profile to steepen into shocks?

If you have time, try running the same test in 2D. Be sure to specify the `linear_wave2d` problem generator in the configure steps, and `/tst/2D-hydro/athinput.linear_wave2d` as the input file. This problem generator will produce a series of .ppm images, which you can display as a movie using, for example, the `animate` command in `ImageMagick`. You'll have

to change the maximum and minimum density in the `<output3>` block in the input file to values closer to the extrema produced for this run; try using 2.0 and 1/2 respectively.

4. **Riemann problems (shock tube tests)** Try running the Brio & Wu MHD shock tube test by configuring the code with `configure --with-problem=shkset1d --with-order=3 --with-flux=roe`. Use the input file `/tst/1D-mhd/athinput.brio-wu`. Plot the density, pressure, and all components of the magnetic field and velocity (if you are using IDL, there is a useful procedure in `/vis/idl/pltath.pro` for this task). Then try running the same test with 1st order and 2nd order reconstruction (remember you'll have to reconfigure and recompile the code for each test). Plot the density at the final time for each order of reconstruction (first, second, and third) on the same plot and compare.

Then try running the Brio & Wu shocktube in 2D. In this case, the 1D planar shock front propagates at an oblique angle to the grid. Configure with `configure --with-order=3 --with-flux=roe --with-problem=shkset2d`. You'll have to edit the 1D input file to use a 2D domain of size 1×0.5 and to use $N \times N/2$ grid cells. You probably won't have the patience to use $N = 800$, but using the largest resolution you can and try comparing the profile in an $x-$slice from the 2D data to the 1D result. Note how much more complicated the source code for Riemann problems run in 2D is compared to the 1D case. Keeping $\nabla \cdot \mathbf{B} = 0$ is trivial in 1D, but is a serious issue in multidimensions. Running planar shock tube tests at an oblique angle in 2D such as the above helps investigate such issues, but it requires $\nabla \cdot \mathbf{B} = 0$ in the initial conditions, which makes the problem generator non-trivial. By the way, the field loops advection tests described in the Athena method papers (run by configuring with `--with-problem=field_loop`) are a much better test of whether the MHD algorithm preserves the divergence-free constraint. In particular, if a transverse velocity is added along the axis of the field loop, the out of plane component of $\mathbf{B}$ must remain zero. For reasons described in the method papers, this is quite difficult to achieve numerically, although the CT algorithm in Athena is able to pass this test.

5. **Shocks in multidimensions.** You can learn much from watching the evolution of shocks in multidimensions. One interesting problem is the double Mach reflection test shown in the lectures (you can run this test using the `dmr.c` problem generator, and the `/tst/2D-hydro/athinput.dmr` input file). However, we will study shock-cloud interaction in 2D hydrodynamics instead. This problem consists of a Mach 10 planar shock which overruns a sphere (actually, an infinite cylinder in 2D) with a density 10 times larger then the background. Configure the code with
`configure --with-order=3 --with-flux=roe --with-gas=hydro --with-problem=shk_cloud`.
Run the problem using the `/tst/2D-hydro/athinput.shk_cloud`. Animate the resulting density (.ppm) images and see what happens. Try increasing the resolution as much as you can afford and see how this affects the dynamics, and dump movies of the other variables (especially useful is the pressure) by adding more `<output>` blocks to the input file using the .ppm image format. At early times, you should be able to identify the reflected and transmitted shocks near the leading edge of the clouds. At later times, the incident shock refracts around the cloud and interacts with itself, producing strong vorticity. At even later times the cloud is shredded by instabilities as it is advected off the grid. See if you can spot all these features.

6. **Kelvin-Helmholtz instability** Our final homeowork problem is the nonlinear evolution of a classic fluid dynamical instability, the KHI. It is produced by certain types of shear layers in a fluid. Often the instability is studied for a slip-surface (discontinuous shear layer), for example see the tests on the Athena web pages. However, in this case there is no length scale in the problem, and the fastest growing modes are at the grid spacing. Instead, we will study the KHI in a shear layer of finite (resolved) width, using a single mode perturbation. Configure the code with `configure --with-problem=kh --with-order=3 --with-flux=roe --with-nscalars=1`, and use the input file `/tst/2D-mhd/athinput.kh`. The last configure option adds one passive scalar to the evolution, which can be used to track the two layers. Running the code will produce images of two variables, the transverse velocity and this scalar variable (labelled "C"). Watch movies of these images to follow the growth and nonlinear saturation of the instability. To be more quantitative, plot the time evolution of the logarithm of the kinetic energy in the transverse component of the velocity (which is dumped in the "history" file, `kh.hst`). Use the slope of the line at early times to measure the growth rate in the linear regime. Does it agree with analytic theory?

7. **Applications.** It is very easy to run Athena in parallel by adding the `--enable-mpi` option to the configure step, and by adding the appropriate paths to the MPI library installed on your machine using the `Makeoptions.in` file. As time permits, feel free to play with running larger versions of the above problems in 3D using MPI and any parallel cluster to which you have access. You'll have to add a `<parallel>` block to the input file you use to control the MPI decomposition. There are many problem generators and input files to try, but one good one is the shock-cloud test we ran above. The User's Manual can give you hints on tools to look at the data in 3D (for example, VisIt). This problem is only for those who are interested and want to move beyond 2D tests run on serial processors.