For these two short assignments, you will be writing a small C++ program to implement a simulator. Some people will implement an event-oriented simulator and some will implement a levelized-compiled-code simulator. To simplify your code, neither of these need handle edge-triggered flops. You do need to handle level-sensitive latches, but the network will not require any iteration in a single phase.

To avoid reinventing the wheel, you should start with our network reader. It is documented in /comp/150CAD/public_html/HWs/pdf/network_reader.pdf . You should start with the file sim_event.cxx (for an event-oriented simulator) or sim_lcc.cxx (for levelized compiled code) and add your own routines to it. Both of these files can be found in /comp/150CAD/public_html/HWs/code. When you're done, turn in just the file sim_event.cxx or sim_lcc.cxx via the Provide interface.

For both simulators, you should use two networks (from /comp/150CAD/public_html/HWs/gmf):

1. The same small network as in the paper-and-pencil homework (from the file HW1_network1.gmf). This is purely combinatorial logic. Note that the paper-and-pencil homework shows inputs rising or falling at t=0. To mimic this reasonably closely, you should set a phase length of 20; your network supplies two input vectors, and the second vector thus launches the rising or falling transition at t=20 (rather than at t=0).
2. A two-bit counter (in HW1_counter_2bit.gmf). This is a small state machine built from level-sensitive latches.

Your code should run the simulation, compute the output vectors and print the results. Furthermore:

- if you are doing the discrete-event simulator, you will likely have to make sure your code simultaneously processes *all* events that occur at the same virtual time, as we showed in the metacode in class. You will also likely need to initialize your network (as we hinted at in class). If you find that you do not get the right answers, then you should first try to isolate the issues yourself and then we can discuss them in class.
- feel free to compare your answers with other students, even (or especially) with other students who are solving the problems manually.