Problem 0:  Follow the instructions for account setup and homework workflow on the course procedures page, [http://www.ece.lsu.edu/koppel/v/proc.html](http://www.ece.lsu.edu/koppel/v/proc.html). Run the testbench on the unmodified file. There should be errors on all but the min_4 (Four-element) module. Try modifying min_4 so that it simulates but produces the wrong answer. Re-run the simulator and verify that it’s broken. Then fix it.

Note: There are no points for this problem.

Problem 1:  Module min_n has an elt_bits-bit output elt_min and an elt_count-element array of elt_bits-bit elements, elts. Complete min_n so that elt_min is set to the minimum of the elements in elts, interpreting the elements as unsigned integers. Do so using a linear connection of min_2 modules instantiated with a genvar loop. (A linear connection means that the output of instance i is connected to the input of instance i + 1.)

Verify correct functioning using the testbench.

Problem 2:  Module min_t is to have the same functionality as min_n. Complete min_t so that it recursively instantiates itself down to some minimum size. The actual comparison should be done by a min_2 module.

Verify correct functioning using the testbench.

Problem 3:  By default the synthesis script will synthesize each module for two array sizes, four elements and eight elements.

(a) Run the synthesis script unmodified. Use the command rc -files syn.tcl. Explain the differences in performance between the different modules.

(b) Modify and re-run the synthesis script so that it synthesizes the modules with elt_bits set to 1.

The synthesis program should do a better job on the behavioral and linear models. Why do you think that is? Hint: The 1-bit minimum module is equivalent to another common logic component that the synthesis program can handle well.