For this assignment, you will be writing a small Matlab program to implement an inverter-sizing tool. The tool will:

- use a library routine to create a cell library of available inverters.
- take an output load and an input driver specified by the user
- just like foils #21-29 from the gate-sizing lectures, use dynamic programming to pick the cells that result in the minimum delay from input to output.

You can get the file sizing.m from /comp/150CAD/public_html/HWs/code. It contains a main() function and skeletons of the functions you are to write. It also contains two small functions that may be useful:

- dump_dp (n_stages): prints out the dynamic program that you've constructed. You can use this to debug your dynamic program while building it, or to check your answer when you're done.
- delay (cell, load). Calculates the delay of the given library cell when it sees the given load. 'Cell' should be a number from 1-4; 'load' is the load in ff.

You have the skeletons of two functions:

- create_dp (n_stages, Cout). This creates the dynamic program just as we did in class.
- execute_dp (n_stages, in_cell). This should print out the optimal inverter chain that you've discovered, with the sizes and gate delays for each of the inverters you've chosen.

There is one trick to make the coding cleaner. While we only worked with 3 cells in class (W_n=300, 100 and 30), we also need the known input driver (W_n=10). Thus, you are given all four cells in the library. The easiest way for make your code clean is:

- At each stage, choose from all four available cells. Thus, your dynamic-program table will have 4 columns and not 3.
- When building the dynamic program, treat the bottom row just like the other rows. I.e., fill in all four columns.
- When you execute the dynamic program, just start with stages[1,4], which is the W_n=10 cell. I.e., you have filled in stages[1,1], [1,2] and [1,3] for no reason other than it makes your code a bit cleaner; you never look at those numbers.

One hint: this is exactly the same problem that we did in class, with the same C_{load} and the same available inverter cells. Thus, the final answer (both the delay and pointers tables and the final choice of inverters) should be just what we discovered in class.

When you're done, turn in just the file sizing.m via the Provide interface.