Stata Hints:

In exercise 15.1 you have to plot the values of the lag weights against time. In Stata this is pretty simple to do using loops. Here is how I do it:

1. Load the data and make sure that it is set up as a time series.
2. Use the summarize command to get summary statistics for the desired time period. You can use the `if tin(start_date, end_date)` modifier to adjust the sample.
3. Compute bandwidth. This is used with the `newey` command later in the exercise.
4. Put the value of the first multiplier into your dataset.

```
generate mult =_b[x] in 1
```

   This assumes that the variable name for the contemporaneous value of x is `x`. You could also initiate this with a missing value.

5. Use a `forvalues` loop. Here's a little bit about the basic structure of a loop.

Loops can make model selection and other repetitive computations much easier. As an example, let's estimate a series of distributed lag models over all possible models for `q=1, 2, 3, 4, 5`. The basic structure would be

```
forvalues q=1/5 {
        [statements to compute and print]
    }
```

The loop is executed as long as the value of `q` is within the given range (e.g., between 0 and 5 inclusive for `q`). In this form the value of `q` will increment in steps of 1.

Braces must be specified with `forvalues`, and

i. the open brace { must appear on the same line as `forvalues`;
ii. nothing may follow the open brace except, of course, comments; the first command to be executed must appear on a new line;
iii. the close brace } must appear on a line by itself.

So, to estimate a series of distributed lag models using data for third quarter of 1988 and beyond (not what you are doing in the assignment, by the way), use the statement:

```
forvalues q=1/6 {
      quietly regress y L(0/`q').x if date >= tq(1988q3)
      }
```

Notice a couple of things about the statements that are being computed within the loops. First, `q` is now referred to by its macro name. That means that it must be enclosed in single quotes (left and right as we did above). Second, the lag and difference operators can be included in one statement, `L(0/`q').x`. As `q` increments from 1 to 6, lags are added to the model.

The loop can also be used to pick out coefficients from the distributed lag (multipliers).

```
forvalues q=0/6 {
local  I = `q'+1
quietly replace mult =_b[L`q'.x] in `I'
}
```

You can also compute upper and lower confidence bounds within the loop. Generate a lag variable that runs from 0 to 6.

```
gen lag = _n-1 in 1/7
```

When this example loop runs, it will fill up the first 11 rows of the data matrix with multipliers—impact through the 6 period delay multiplier. Us the in modifier to print out or graph what you want.

```
list mult in 1/7
line mult lag in 1/7
```