The comma operator

- See `test_comma_op.c` created in class to see what would happen if you use the comma operator in expr2 of the loops. Note how the warning tells you that the left expression of the comma would be ignored, and that the loop eventually only depends on the right expression after the comma. Do not use commas in this way because of these problems. While the syntax is correct, it is probably doing something other than you want it to do -- hence the warning produced when you compile.

Print squares using different loops

- See `square-while.c` , `square-do-while.c` , and `square-for-loop.c` for different variations of the solution.

Trace output

- See `nested-loop-break.c` for the example on Slide 13. Try to trace the output yourself and then compare it with running the program.

Print a calendar while accepting program arguments

- See `calendar.c` for the solution developed in class. Note that my program does not behave correctly if the starting day is 7. Can you spot the error? Try to debug the program and see how you can fix this behavior.
- Note that you may also see the book solution here http://knking.com/books/c2/answers/c6.html
- Note that I do not handle the case where an invalid argument is passed in (i.e., some value that is not a number like "abc"). Check the `atoi` function documentation to see what the function returns when it cannot convert the given string (you can google "atoi C function").

Extra programs

These sample programs have not been demoed in class

- Check break-continue.c .. Try running the program once with the break statement there and once with changing the break; to continue;
- Check convert.c to see how to convert a number read as a string to an int. Remember that we have already seen the atoi function on Slide 17 when working on the calendar.c program.

Extra files

- Check the Makefile I have there. We have a video tutorial on Makefiles that you can watch. The Makefile included here has two targets, one for the convert.c program above and one for the break-continue.c program above.
    - This means that instead of typing `gcc -Wall -std=c99 -o convert convert.c` to compile the convert.c program, I can simply type make convert since the name of the target I want to execute (i.e., that name before the `:`) `is convert`
    - Similarly, instead of typing `gcc -Wall -std=c99 -o break-continue break-continue.c` to compile the `break-continue.c` program, I can simply type `make break-continue` since the name of the target I want to execute (i.e., that name before the `:`) is `break-continue`
    - See the `clean` target? It simply removes all temporary files or executable files that got generated using the `rm` command.