# Courier Final Demo

**Description:**

This demo will explain how to build Courier and run Courier programs. It will highlight a few more interesting test-cases, as well as feature instructions on how to run and play around with the test suite.

My Project includes a few things. A handful of .ml, mly, .mll, .mli, and makefile files. These are the code itself. There are some files with extensions like .cmo, these can be ignored. It then includes a tests directory, which includes a variety of .cr test files. It includes a demo directory, which includes some .cr tests, this document, and a blank demo .cr file meant for quick coding. The Zip file comes 'pre-built', but I include the instructions to build it if desired.

**Instructions for building/running the project:**

1. All Commands mentioned should be ran in the terminal
2. Unzip Courier.zip(if this isn't done yet)
3. Cd into the Courier directory. Here you should see the tests directory, an eval.ml file, main.ml, etc.
4. If you want to continue with the prebuilt project and have not yet deleted anything, you can skip this step
    a. Run "make clean" to ensure everything is clear
    b. Ocamlbuild -use-menhir -no-hygiene parser.ml
    c. ocamlbuild -no-hygiene lexer.ml
    d. Move parser.ml, parser.mli, and lexer.ml from the _build directory into the Courier directory(i.e. bring them one directory up so they are in the same directory as ast.ml, eval.ml, etc)
5. Run "make" to build the files
6. Now files can be ran in the terminal with: ./cr <name of file>

**Things to do in my Demo:**

- Run some of the generic tests
    - These can be found in the tests directory
    - I'd recommend looking through some of them as a good way to get a hold of the language's syntax
    - Specific recommendations are listed below

- Play around with blank demo.cr file
    - This is a blank file that can be ran quickly with a "make run" command
    - This is a more 'optional' part of my demo
    - Keep in mind that some boolean operators require more parentheses than in other languages

- Run some of the demo programs
    - These are a bit more complex than the generic tests, and hope to demonstrate harder to grasp features.
        - They are implementations of an efficient fibonacci function and a type-agnostic nth function for lists.
    - Can be found in the Demo directory
    - Run using: ./cr demo1.cr or ./cr demo2.cr
    - Demo1
        - A fib function that stores already calculated values in a global list
        - Calculates fibonacci numbers recursively using the Do command
    - Demo2
        - Takes a list and an integer n. If n is zero then return the head of the list, regardless of type.
        - If n is non zero then Match the list with either the empty list(Null list) or non-empty list of any type. If empty then return Null, if non empty then recursively call with the tail of the list and (n-1)

- Look into some of the following tests to see interesting features not touched on in the demo files. More tests on the same/similar topics can usually be found in tests with similar numbers to the numbers below, these are just some I wanted to highlight.
    - Type Handlers: test20.cr, test31.cr
    - Objects/Type Definitions: test25.cr, test26.cr, test31.cr, test58.cr
    - Imports: test38.cr, test60.cr
    - Arbitrary Values: test59.cr, test50.cr, test46.cr, test61.cr