Purpose of this exercise. The purpose of this exercise is to give you basic familiarity with OCaml. You will download and install it; do the tutorial; read a nice recent write-up by Yaron Minsky; read some code for our language implementation; and finally re-do the expanders that you did in Project 2.

You may also use the OCaml installation in `~guttman/bin/ocaml` on ccc.wpi.edu.

Reimplementing the expanders may convince you that the OCaml type declaration, type inference, and compile-time type checking make it vastly easier to manage complex datatypes in OCaml. If you ask me, I think they’re a lot easier, and I made a lot fewer mistakes along the way.

Grading. You will hand in only the last part, the expanders. Each of the four expanders will be worth 25% of the exercise, which is only 5% of your course grade in total.

Download and install OCaml. Start by downloading OCaml (see http://caml.inria.fr/ocaml/index.en.html), and installing it on some convenient computer. Use OCaml 3.12.1, which is the current version. Pre-compiled binaries for several OSs—including Windows MacOS, and several varieties of Linux, and for several architectures—are listed at http://caml.inria.fr/ocaml/release.en.html You can also easily build OCaml from its sources if you have a system set up with make and a C compiler. The commands for a root install are:

```
./configure
make world
make opt
```
sudo make install

If you want to do a non-root install, you can do:

./configure --prefix $HOME
make world
make opt
make install

This will put binaries in $HOME/bin.

Read the first chapter of the manual at http://caml.inria.fr/pub/docs/manual-ocaml/index.html As you read the text, enter the example material into the OCaml interpreter and play around.

Skim chapters 2 and 3.

Now read Yaron Minsky’s interesting article from last September in the ACM Queue at http://queue.acm.org/detail.cfm?id=2038036 There’s also a pointer to it from http://caml.inria.fr/ocaml/index.en.html.

Yaron Minsky has been leading software development at Jane Street Capital, a small Wall Street firm that appears to do all their trading and analysis using OCaml programs that they developed in house. They regularly hire clever, OCaml-literate programmers.

Don’t hand in anything up to this point.

Read the code in the files mast.ml, cmast.ml, especially, and in expand.ml.

In expand.ml, there are four functions that contain a comment and a dummy line. They are the expanders for do-until, for, multiple-argument functions, and multiple-argument applies. Fill them in so as to produce correct cexprs.

If you do that and type make, you will obtain an executable that you can use to parse programs in .txt form, expand their operators, and write them to expanded .lua format. Check that you’re getting correct output.

You can also load the compiled source files into the REPL by typing

#use "load_me.ml";;

(including the #; it’s not the prompt) to the running OCaml interpreter.

Please hand in your version of expand.ml and your test files, with a README saying what the test files demonstrate, via turnin at URL https://turnin.cs.wpi.edu:8088/turnin.rkt.