CSE4102: Programming Languages

Homework 0: Overview and OCaml Setup

Prof. Stefan Muller
TA: Ricky Cheng

Out: Thursday, Jan. 22
Due: Monday, Feb. 2 11:59pm EST

Updated Jan. 29

This assignment contains 1 programming task for a total of 10 points.

1 Logistics

Please read and follow these instructions carefully.

Collaboration and Academic Honesty. This homework is to be completed individually. See the syllabus for more detail on the collaboration policy, and in particular the policy on AI.

2 Introduction

Welcome to the course! As mentioned in class, we will be learning functional programming for the next few weeks. Before we get started, we need to set up our programming environment. In particular, we will need to install the following tools:

- OCaml: A static, strongly-typed functional programming language. It is widely used in industry (e.g., at Jane Street, Facebook) and academia because of its powerful type system.
- opam: Opam is the official package manager for OCaml. You can think of this as the OCaml equivalent of `pip` (Python) or `npm` (JavaScript) with virtual environments. Opam helps us install the compiler and manage external libraries. As such, it is the preferred way to install OCaml.

3 Installing OCaml

As you have seen in class, OCaml has a great online playground to try out the language. However, interactive environments become a hindrance once projects get large. So it is best to use interactive environments only for quick prototyping and use the compiler for projects.

Your first task is to get OCaml running on your machine. The OCaml community maintains an excellent, cross-platform installation guide.

1. Go to the OCaml Installation page.
2. Choose your platform (Unix or Windows) and follow the instructions for installation.
   Updated 1/29: Note: If you install natively on Windows, your instructions for the rest of this homework (and future homeworks) will be a little different. That's fine; we'll try to make sure to always give both

sets of instructions and you can let us know if you ever have questions or problems. Another option is to install using WSL2 using these instructions. Then the Linux/Mac instructions for the rest of the OCaml installation and for running OCaml programs should work for you.

3. Once you have finished installing, open up a new terminal and enter `ocaml -version`. You should see something similar to "`The OCaml toplevel, version 5.X.X`". This means you have OCaml installed!

4 Assignment 0

This assignment is designed to ensure that you have the environment set up properly. There are *Updated 1/29*: three files provided: `hw0.ml`, which is the code file you will be editing and submitting, a `Makefile` *Updated 1/29*: (Linux/Mac/WSL2) that compiles the code into an executable binary, *Updated 1/29*: and a `make.cmd` script (Windows) that will do roughly the same thing as the Makefile.

1. Copy the appropriate files for your platform into a clean directory from HuskyCT.
2. Open `hw0.ml`, enter your name at the top. There are five true/false questions in the comments, answer them by replacing the expressions below from `raise Unimplemented` with either `true` or `false`.
3. When you are done editing the program:
   - *Updated 1/29*: On Linux/Mac/WSL2: run `make`, which creates an executable named `hw0` (if using WSL2 on Windows, make sure you've opened a shell for whatever Linux distro you installed and navigated to the right folder).
   - *Updated 1/29*: On native Windows: run `make.cmd`, which creates an executable named `hw0.exe`
4. *Updated 1/29*: On Linux/Mac/WSL2: Run the executable `./hw0`, which outputs a file named `hw0_out`.
5. *Updated 1/29*: On native Windows: Run the executable `hw0.exe`, which outputs a file named `hw0_out`.

4.1 Submission Details

For this class, we will use Gradescope to give feedback on both the coding and written assignments. For coding assignments, an autograder will be set up so you can get quick feedback on your program.

For this submission, you will need to turn in the following two files on Gradescope:

- `hw0.ml`
- `hw0_out`

Note: Please ensure that the files are named as above, and make sure to submit both files! (One of the automated tests that runs after you submit will fail if you forget to include `hw0_out`.) Additionally, please submit the files as individual files and not as a single zip file.

After you submit, you'll see results of some of the autograder tests. These do some basic checks that the required files are there and compile correctly. **If your .ml file doesn't compile, you will not get credit for the homework.** They may also do some basic tests on the correctness of your code, but we may also run other correctness tests whose results are not shown to you until after the deadline, so it's important to always test your code yourself (on future assignments, not so much on this one.)

Gradescope allows two methods of submission: a file upload or a submission through git via GitHub or BitBucket.

4.1.1 File Upload

If you are submitting the files directly, select "Upload" to upload the two files directly.

4.1.2 Git Submission

To submit via GitHub or BitBucket, you'll need to connect your account. Then you will need to choose the repository and branch that your assignment is on. **Please ensure that the two files are in the top-level directory of the repository**. This is because Gradescope clones the entire repository and we do not know what your directory structure may look like.