HW1 – Getting Started with OCaml

CS 476, Fall 2018
Due Sep. 4

1 Instructions

This assignment will help you set up the tools you need for the class, and gauge your knowledge of the basics of functional programming. The programming problems are in Section 3—but read Section 2 first! It will walk you through setting up your OCaml programming environment. Please don't hesitate to ask for help on Piazza (`https://piazza.com/class/jkh8q52qrh06v`).

You can submit this assignment through Gradescope (`https://www.gradescope.com`).

2 Getting Started with OCaml

OCaml is a functional programming language similar to F#. You can find downloads and documentation at `https://ocaml.org`. I recommend installing the OPAM package manager, and using Emacs or Vim with OCaml extensions to write your code. If you run into any problems getting started, you can ask for help on Piazza at `https://piazza.com/class/jkh8q52qrh06v`. Here are the steps to getting OCaml running on your computer:

1. Install the OPAM package manager.

   On Windows: go to `https://fdopen.github.io/opam-repository-mingw/installation/` and download and install the environment. This will give you a new installation of Cygwin (Unix-style terminal for Windows) with OPAM installed.

   On all other platforms: go to `https://ocaml.org/docs/install.html` and find the instructions specific to your package manager.

2. Get OCaml extensions for your preferred text editor, as many as you like. I recommend Tuareg (Emacs only) and Merlin (Emacs and Vim). (If you find any other useful extensions, let the class know on Piazza!) You can install them by running, e.g., `opam install tuareg` on the command line. If any more configuration is needed, it will be mentioned in the output of `opam`.

3. Try running some OCaml code! You can run `ocaml` on the command line to start the interactive read-eval-print loop (REPL). At the `#` prompt, you can type a line

of code ending in `;;`, and the REPL will display the results of executing that code. Try reproducing the following session:

```
# print_string;;
- : string -> unit = <fun>
# print_string "Hello world\n";;
Hello world
- : unit = ()
# #quit;;
```

The OCaml compiler is called `ocamlc`, and takes files with the extension `.ml`. Try writing some code in `test.ml` (such as `print_string "Hello world\n";;`) and then running:

```
ocamlc -o test test.ml
./test
```

3 Writing OCaml Functions

To receive full credit for the following problems, make sure your code compiles, and *do not* use mutable references (i.e., the `ref` keyword).

1. (3 points) In class, we defined a type `intlist` of lists of integers as

   ```
   type intlist = Nil | Cons of int * intlist
   ```

   Write a function `is_nil : intlist -> bool` that is `true` for `Nil` and `false` for any non-`Nil` list. For instance, `is_nil Nil` should be `true`, and `is_nil (Cons (1, Nil))` should be `false`.

2. (3 points) Write a function `sum : intlist -> int` that returns the sum of all the elements of an `intlist`. For instance, `sum Nil` should be 0, and `sum (Cons (2, Cons (3, Nil)))` should be 5.

3. (4 points) Define a type `int_or_list` that has two constructors: `Int`, which takes an `int`, and `List`, which takes an `intlist`. Write a function `is_pos : int_or_list -> bool` that is `true` for any positive `int` and any list whose `sum` (as defined in problem 2) is positive, and `false` otherwise. (A number is positive if it is greater than 0.) For instance, `is_pos (Int 0)` and `is_pos (List Nil)` should be `false`, and `is_pos (Int 3)` and `is_pos (List (Cons (1, Nil)))` should be `true`.