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/jushbmjmi13yb?cid=9).

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 Visual Studio Code with an OCaml extension to write and test your code. If you run into any problems getting started, you can ask for help on Piazza at https://piazza.com/class/jushbmjmi13yb?cid=9. Here are the steps to getting OCaml running on your computer:

1. Install OCaml and the OPAM package manager by going to https://ocaml.org/docs/install.html and finding the instructions specific to your OS/package manager.

2. If you want to use Visual Studio Code, open it up and install the OCaml extension. If you want to use Emacs or Vim, use OPAM to install the extensions Merlin, ocp-indent, and Tuareg (Emacs only). You can install them by running, e.g., opam install merlin 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 start the interactive read-eval-print loop (REPL) by running ocaml or rlwrap ocaml on the command line, or pressing Ctrl+Shift+P and choosing “OCaml: New REPL Session” in VSCode. 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:
3 Writing OCaml Functions

Begin by downloading the file `hw1-base.ml` from the course website, and renaming it to `hw1.ml`. This file contains the outline of the homework, with incomplete definitions commented out. For each problem, remove the comments and fill in the associated definitions. You can also add functions and modify the outline as you see fit.

To receive full credit, make sure you don’t get any errors when running `#use "hw1.ml"`, and do not use mutable references (i.e., the `ref` keyword).

1. (2 points) Write a function `fact` that takes an `int` $n$ and returns the factorial of $n$. You should be able to produce the following session:

```ocaml
# fact;;
- : int -> int = <fun>
# fact 5;;
- : int = 120
```

2. (3 points) Write a function `has_key` that takes a map (of type `string -> string`) and a key of type `string` and returns `true` if the map has the key (i.e., if it maps the key to a value other than `""`), and `false` otherwise. For instance, `has_key map1 "a"` should return `true`, and `has_key empty_map s` should return `false` for any `s`.

3. (4 points) Write a function `set_to_bag` that takes a set (of type `string -> bool`) and returns a bag (type `string -> int`) that has exactly 1 of each element in the set. For instance, `set_to_bag set1` should return a bag $g$ such that `bag_lookup $g$ s` returns 1 if $s$ is "a" or "b", and 0 otherwise.

4. (2 points) In class, we defined a type `intlist` of lists of integers as

```ocaml
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
5. (3 points) Write a function \texttt{product} : \texttt{intlist} $\rightarrow$ \texttt{int} that returns the product of all the elements of an \texttt{intlist}. For instance, \texttt{product Nil} should be 1, and \texttt{product (Cons (2, Cons (3, Nil))))) should be 6.

6. (4 points) Define a type \texttt{string_or_list} that has two constructors: \texttt{String}, which takes a \texttt{string}, and \texttt{List}, which takes an \texttt{intlist}. Write a function \texttt{is_empty} : \texttt{string_or_list} $\rightarrow$ \texttt{bool} that is \texttt{true} of an empty string or a list with no elements, and \texttt{false} otherwise. For instance, \texttt{is_empty (String "")} and \texttt{is_empty (List Nil)} should be \texttt{true}, and \texttt{is_empty (String "hi")} and \texttt{is_empty (List (Cons (1, Nil)))} should be \texttt{false}. 
