Programming Languages (COP 4020/CIS 6930) [Fall 2014]

Assignment II

Objectives

1. To understand and use higher-order functions, including map and fold.
2. To continue practicing writing functional programs, using ML features like Currying.

Due Date: Sunday, September 21, 2014 (at 11:59pm).

Machine Details: Complete this assignment by yourself on the following CSEE network computers: c4lab01, c4lab02, ..., c4lab20. These machines are physically located in the Center 4 lab (ENB 220). Do not use any server machines like grad, babbage, sunblast, etc. You can connect to the C4 machines from home using SSH. (Example: Host name: *c4lab01.csee.usf.edu* Login ID and Password: <your NetID username and password>) You are responsible for ensuring that your programs compile and execute properly on these machines.

Assignment Description

(0) Read Sections 5.1, 5.3-5.6, and 6.1-6.3 of the *Elements of ML Programming* textbook.

(1) Create a file called *as2.sml*. In that file, re-implement all the undergraduate-level functions from Assignment I (i.e., `cadd`, `padd`, `smult`, `pmult`, `ppderivative`, and `peval`), without defining any extra top-level functions. For this second assignment, your functions must:

a) Never call built-in/library functions, *except* `map`, `List.tabulate`, `foldr`, and `foldl`, which are documented at: http://www.cs.princeton.edu/~appel/smlnj/basis/list.html
b) Never be recursive. All recursion must occur within `map`, `List.tabulate`, and `fold`s.
c) Never cause side effects (such as I/O or pointer/array operations) to occur.
d) Match the types given below (*note that arguments must be Curried*).
e) Be commented and formatted properly (again please use spaces instead of tabs); as part of this restriction, limit line widths to 100 characters.
f) Use ML constructs like pattern matching and let-environments when appropriate.
g) Compile and execute on the C4 machines with no errors or warnings.
h) Not be significantly more complicated than necessary.
i) Be reasonably efficient.

As on Assignment I, you can assume that (1) all polynomials passed as arguments to *as2.sml* functions lack negative exponents, and (2) all variable numbers passed as arguments to `ppderivative` are positive.

Function Types

```
- use "as2.sml";
[opening as2.sml]
[autoloading]
[library $SMLNJ-BASIS/basis.cm is stable]
[autoloading done]
val cadd = fn : int -> int list list -> int list list
val padd = fn : int list list -> int list list -> int list list
val smult = fn : int -> int list list -> int list list
val pmult = fn : int list list -> int list list -> int list list
val ppderivative = fn : int -> int list list -> int list list
val peval = fn : int list list -> int list -> int
val it = () : unit
- (* The Assignment I handout provides sample executions for these functions.*)
  (* Because cf (canonical forms) isn't implemented for Assignment II, your  *)
  (* as2.sml functions don't need to output polynomials in canonical form.   *)
```

As always, we will test submissions on inputs not shown in the assignment handouts.

Hints

My *as2.sml* is 41 lines of code (not counting comments/whitespace) and took me 2-3 hours to write and test.

`List.tabulate` can help implement exponentiation in `peval`.

Submission Notes

- Type the following pledge as an initial comment in your *as2.sml* file: "I pledge my Honor that I have not cheated, and will not cheat, on this assignment." Type your name after the pledge. Not including this pledge will lower your grade 50%.
- Upload and submit your *as2.sml* file in Canvas.
- You may submit your assignment in Canvas as many times as you like; we will grade your latest submission.
- For every day that your assignment is late (up to 3 days), your grade reduces 10%.