Programming Languages (COP 4020/6021) [Spring 2017]

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.
3. To gain experience with deductive systems, inference rules, and proving properties of judgments by induction on their derivations.

Due Date: Monday, February 6, 2017 (at 5pm).
Late submission: You may submit any part (or both parts) of this assignment late (i.e., between 5pm on 2/6 and 5pm on 2/8) with a 15% penalty on the whole assignment.

Machine Details: Complete this assignment by yourself, the programming portion on the following CSEE network computers: c4lab01, c4lab02, ..., c4lab20. These machines are physically located in 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) **Programming portion:** Create a file called *xy2.sml*. In that file, re-implement the functions `evalxy`, `multxy`, `paddxy`, and `pmultxy` from Assignment I, without defining any extra top-level functions. Note: The `pmultxy` function is worth 10% extra credit for undergraduates.

For this second assignment, your functions must:

a) Never call built-in/library functions, *except* `map`, `foldr`, and `foldl`.
b) Never be recursive. All recursion must occur within `map` and `fold`s.
c) Never contain let- or local-expressions. The keyword “let” (and similarly, “local”) should never appear in *xy2.sml*.
d) Never cause side effects (such as I/O or pointer/array operations) to occur.
e) Match the types shown below (*note that arguments must be Curried*), without defining any extra top-level functions.
f) Be commented and formatted properly; as part of this restriction, use spaces instead of tabs and limit line widths to 100 characters.
g) Use ML constructs like pattern matching and anonymous variables when appropriate.
h) Compile and execute on the C4 machines with no errors or warnings.
i) Not be significantly more complicated than necessary.
j) Be reasonably efficient (technically, the `evalxy`, `multxy`, `paddxy` functions must run in linear time, and the `pmultxy` in worst-case quadratic time). Full-credit submissions will not contain the @ operator.

Function Types
```
val evalxy = fn : int list list -> int -> int -> int
val multxy = fn : int list list -> int -> int list list
val paddxy = fn : int list list -> int list list -> int list list
val pmultxy = fn : int list list -> int list list -> int list list
```

Hints
My *xy2.sml* is 46 lines of code (not counting comments/whitespace) and took me 3-4 hours to write and test.

To select the i^{th} element of a tuple T, use `#i T`. For example, `#2(3,4,5,6)` returns 4. This selection operation may be useful if, for example, you want to fold through a list to build a tuple, but then just return one element of the final tuple value.

Later functions may call earlier functions; e.g., `multxy` could invoke `evalxy`.

You can test your implementations on the examples shown in the Assignment I handout, though for this assignment arguments need to be curried. As always, we'll test your submissions on examples not shown in the handouts.

Programming-portion Submission Reminders
- Type the following pledge as an initial comment in your *xy2.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 *xy2.sml* file in Canvas.
- You may submit your assignment in Canvas as many times as you like; we'll grade your latest submission.

(2) **Theory portion:**
Recall the Z and S rules from class, used to define natural numbers. E.g., Z nat is derivable using the Z rule, and S(S(Z)) nat is derivable using the S rule twice and then the Z rule. For the remainder of this assignment, assume that every use of the symbol N identifies a natural number, so implicitly, N nat is derivable. Now let's define the PZ and PS rules for addition as follows.

$$\boxed{N_1+N_2=N_3} \qquad \frac{}{N+Z=N}\ \text{PZ} \qquad \frac{N_1+N_2'=N_3'}{N_1+S(N_2')=S(N_3')}\ \text{PS}$$

Using this deductive system, prove that addition is commutative.
Theorem. For all N_1, N_2, and N_3: if $N_1 + N_2 = N_3$ then $N_2 + N_1 = N_3$.

Grading Notes
Partial credit is always possible. If you get stuck, just explain informally whatever ideas you're having trouble stating formally.

Theory-portion Submission Reminders
- Write the following pledge at the end of your submission: "I pledge my Honor that I have not cheated, and will not cheat, on this assignment." Sign your name after the pledge. Not including this pledge will lower your grade 50%.
- For full credit, turn in a hardcopy (handwritten or printed) version of your solutions.
- Late submissions may be emailed or submitted in hardcopy.
- All emailed submissions, even if sent before the deadline, will be graded as if they were submitted late, i.e., with a 15% penalty.

- If you think there’s a chance you’ll be absent or late for class on the date this assignment is due, you’re welcome to submit solutions early by giving them to me or the TA before or after class, or during any of our office hours.