# **Programming Languages (COP 4020/6021) [Spring 2019]**
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 4, 2019 (at 5pm).
**Late submission:** You may submit any part (or both parts) of this assignment late (i.e., between 5pm on 2/4 and 5pm on 2/6) 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, ..., c4lab19. These machines are physically located in ENB 216. 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 *as2.sml*. In that file, re-implement all the undergraduate-level functions from Assignment I (i.e., `constplus`, `polyplus`, `constmult`, `polymult`, `partialderivative`, and `polyeval`), 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://sml-family.org/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 `partialderivative` are positive.

*Function Types*
```
- use "as2.sml";
[opening as2.sml]
[autoloading]
[library $SMLNJ-BASIS/basis.cm is stable]
[autoloading done]
val constplus = fn : int -> int list list -> int list list
val polyplus = fn : int list list -> int list list -> int list list
val constmult = fn : int -> int list list -> int list list
val polymult = fn : int list list -> int list list -> int list list
val partialderivative = fn : int -> int list list -> int list list
val polyeval = fn : int list list -> int list -> int
val it = () : unit
- (* The Assignment I handout provides sample executions for these functions.*)
  (* Because c (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` may be helpful for implementing exponentiation in `polyeval`.

*Programming-portion Submission Reminders*

- 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'll grade your latest submission.

(2) **Theory portion:**
(a) Define a deductive system having two judgment forms:

i. Judgments of the form *N Nat* are valid iff N is a natural number. Just use the same 2 inference rules we discussed in class.
ii. Judgments of the form $N_1$ - $N_2$ = $N_3$ are valid iff subtracting natural-number $N_2$ from natural-number $N_1$ produces natural-number $N_3$. For example, S(S(S(Z)))-S(S(Z))=S(Z) should be derivable, but for all N, S(S(Z))-S(S(S(Z)))=N should not be derivable. Your rules for subtraction judgments can implicitly assume that all numbers involved are natural numbers; your rules for subtraction judgments therefore don't have to contain judgments of the form *N nat*. (Essentially, we're assuming that, in subtraction judgments, the symbol N always refers to a valid natural number.)

(2) Using your definitions from Step (1), formally prove the following Lemma A.

Lemma A. $\forall$ N: (N nat $\Rightarrow$ N-N=Z)

(3) [This step is for graduate students; undergrads may complete this step for +5% extra credit] Again using your definitions from Step (1), formally prove the following Lemma B.

Lemma B. $\forall N_1, N_2, N_3$: $(N_1\text{-}S(N_2)=N_3 \Rightarrow N_1\text{-}N_2=S(N_3))$

(4) Using your definitions from Step (1), formally prove the following Theorem C.

Theorem C. $\forall N_1, N_2, N_3$: $(N_1\text{-}N_2=N_3 \Rightarrow N_1\text{-}N_3=N_2)$

If helpful, your proof of Theorem C may assume that Lemmas A and B hold.

*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.