Programming Languages (Coursera / University of Washington)
Assignment 0

This "assignment" is **entirely optional**. It will have no impact on your grade, even if you do not turn it in at all.

The purpose of the "fake assignment" is only to introduce you to the *mechanics* of completing and turning in assignments — to let you practice with the sometimes awkward interface before completing Homework 1. It looks like the real assignments in the course except the actual content is silly. The content will make more sense after watching the first few videos of Section 1, but you are welcome to turn in this "fake assignment" before starting Section 1.

Download `hw0provided.sml` from the course website. The provided SML code defines three functions. There is only one problem in this assignment:

1. The provided function `f` uses addition where it should use multiplication. In the line defining the function `f`, replace the `+` with `*` to fix this bug.

To test your work, you can start with the provided file `hw0test.sml` and add more tests, or you can make your own testing file. The material in Section 1 explains how to write and use test files. The page on installing and using SML and Emacs is also helpful. Of course, this "fake assignment" is so simple, you can try turning it in without writing or running tests, which is definitely a bad idea for the real assignments. For the real assignments, the provided tests are *very* minimal, designed only to help you understand the necessary format of the solution.

Summary

Evaluating a correct homework solution should generate these bindings (but for this "fake assignment" the provided code will generate these bindings whether or not you fix the bug):

```
val f = fn : int * int -> int
val double = fn : int -> int
val triple = fn : int -> int
```

Of course, generating these bindings does not guarantee that your solutions are correct. *Test your functions: Put your testing code in a separate file. We will not grade the testing file, nor will you turn it in, but surely for real assignments you will want to run your functions and record your test inputs in a file.*

Assessment

We will automatically test your functions on a variety of inputs, including edge cases. We will also ask peers to evaluate your code for simplicity, conciseness, elegance, and good formatting including indentation and line breaks. Your solution will also be checked for using only features discussed so far in class.

Turn-in Instructions

First, follow the instructions on the course website to submit your solution file (not your testing file) for auto-grading. Do not proceed to the peer-assessment submission until you receive a high-enough grade from the auto-grader: Doing peer assessment requires instructions that include a sample solution, so these instructions will be "locked" until you receive high-enough auto-grader score. Then submit your same solution file again for peer assessment and follow the peer-assessment instructions.