HW 1 – Evaluation and Environments

CS 421 – Fall 2014
Revision 1.1

Assigned August 26, 2014
Due September 2, 2014, 23:59 pm
Extension 48 hours (20% penalty)

1 Change Log

1.0 Initial Release.

2 Turn-In Procedure

Answer the problems below, then using your favorite tool(s), save your work as a PDF (either scanned if handwritten or converted from a program), and commit the PDF in your your repositiory in the directory `assignments/hw1`. Your file should be named `hw1-submission.pdf`. The command to commit this file is

```
svn commit -m "Turning in hw1." hw1-submission.pdf
```

3 Objectives and Background

The purpose of this HW is to test your understanding of

- the scope of variables, and the state of environments used during evaluation

Another purpose of HWs is to provide you with experience answering non-programming written questions of the kind you may experience on the midterms and final.

4 Problems

1. (15 pts) Below is a fragment of OCaml code, with various program points indicated by numbers with comments.

```
3 * 7;;
(* 1 *)
let x = 22;;
let y = x / 2;;
(* 2 *)
let z =
  let x = 7 * y
  in
(* 3 *)
  x - y;;
(* 4 *)
let w = if z > x && y < 13 then x + y else z - x
 in
```

```
(* 5 *)
 (2,  w);;
(* 6 *)
let f x = y * x;;
(* 7 *)
let x = f y;;
(* 8 *)
```

For each of program points 1 through 8, please describe the environment in effect after evaluation has reached that point. You may assume that the evaluation begins in an empty environment, and that the environment is cumulative thereafter. The program points are supposed to indicate points at which all complete preceding declarations (including local ones in scope) have been fully evaluated. Answers should be written as a set of bindings (without the use of the update operator).

Solution:

```
3 * 7;;
(* 1 *) { }
let x = 22;;
let y = x / 2;;
(* 2 *) {x → 22, y → 11}
let z =
  let x = 7 * y
  in
(* 3 *) {x → 77, y → 11}
  x - y;;
(* 4 *) {z → 66, x → 22, y → 11}
let w = if z > x && y < 13 then x + y else z - x
 in
(* 5 *) {w → 33, z → 66, x → 22, y → 11}
 (2,  w);;
(* 6 *) {z → 66, x → 22, y → 11}
let f x = y * x;;
(* 7 *)  {f → ⟨x → y * x, {z → 66, x → 22, y → 11}⟩, z → 66, x → 22, y → 11}
let x = f y;;
(* 8 *)  {x → 121, f → ⟨x → y * x, {z → 66, x → 22, y → 11}⟩, z → 66, y → 11}
```