HW 3 – Order of Evaluation

CS 421 – Spring 2014
Revision 1.0

Assigned Thursday, February 6, 2014
Due Sunday, February 16, 2014, 11:59pm

1 Change Log

1.0 Initial Release.

2 Objectives and Background

The purpose of this HW is to test your understanding of:

- Order of evaluation in OCaml

3 Turn-In Procedure

This assignment is named `hw3`. Using your favorite tool(s), you should put your solution in a file named `hw3-solution.pdf`. Your answers to the following questions are to be submitted using the svn repository as described in the section `Instruction for Solving and Submitting Assignments` on the web-page: `http://courses.engr.illinois.edu/cs421/sp2014/mps/index.html`

4 Problems

1. (20 pts) Below is a fragment of Ocaml code. Describe everything that is displayed on the screen (its observable behavior) after each top-level declaration in this code has been cut-and-pasted into an interactive Ocaml session, (followed by a carriage return) and explain why this behavior is observed. This should include both the type information that the compiler gives back for each declaration, and any other things printed to the screen. For the type information, no explanation is required (but it should be correct). Give explanations for all other things printed and the order in which they occur.

   Note: In Ocaml, in the application of an expression of function type to an argument, the arguament is evaluated to a value first, then the expression of function type is evaluated to a functional value. If the functional value is a closure (as opposed to a primitive operation, or a partial application of a primitive operation), then the resulting application of the closure to a value is done as described in class.

   For this problem, we expect, and will accept, an English narrative describing the sequence of computations and branch decisions (e.g. `x > 5` evaluates to `true` since `x` has a value of 12 and $12 > 5$, and therefore we evaluate the `then` branch).

```
let f = (print_string "\na\n";
         fun x ->
          (let r = (print_string "b"; x + 7)
           in if (print_string "c"; x > 0)
              then (print_string "d\n"; 2 * x)
              else (print_string "e\n"; r)
          )
        );;
(* 1 *)

let g y = (print_string "z\n"; y + 2);;
(* 2 *)

let n = g(f(g 0));;
(* 3 *)
```