HW 3 – Order of Evaluation
CS 421 – Fall 2013
Revision 1.0

Assigned Wednesday, September 11, 2013
Due Tuesday, September 17, 2013, 19:59pm
Extension 48 hours (20% penalty)

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 What to `handin`

Answer all parts of the problem below, save your work as a PDF (either scanned if handwritten or converted from a program), and hand in the PDF. The name of this assignment is `hw3`.

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 *)
```