Flolac 2010
Operational Semantics
Assignment 1, Due date: July 1

1. Prove that "`S1; (S2; S3)`" and "`(S1; S2); S3`" are semantically equivalent. Note that one direction of proof is good enough.

2. Specify the semantics of the construct "`repeat S until b`" in the style of natural semantics. The semantics of the repeat-construct is not allowed to rely on the existence of a while-construct in the language.

3. (Bonus) Prove that "`repeat s until b`" and "`s;if b then skip else repeat s until b end`" are semantically equivalent.

Programming Exercise: due date: July 6

Write an interpreter for *While* in OCaml based on the natural semantics of *While*. Note that you need to define an "*exception*" type to model the errors in interpreting *While* programs. Please name your interpreter as `NS`, as shown in the following ode skelton:

```
type num = string
type var = string

type aexp = Num  of num
          | Var  of var
          | Add  of aexp * aexp
          | Mult of aexp * aexp
          | Sub  of aexp * aexp

type bexp = True
          | False
          | ...

type stm  = Ass  of var * aexp
          | Skip
          | Seq  of stm * stm
          | ...

type state = var -> int
```

```
(* n : num -> int *)
let n m = int_of_string m

(* a: aexp -> state -> int *)
let rec a e s = match e with
                   Num m       -> n m
                 | Var x       -> s x
                 | Add (e1, e2) -> a e1 s + a e2 s
                 | ...

type config = Inter of stm * state
          | Final of state

(* state update : to get a new state *)
let update x e s = fun y -> if y=x then a e s else s y

let ns c = match c with
              Inter (Ass (x, e), s) -> Final (update x e s)
            | Inter (Skip,      s) -> Final s
            | ...

exception NotFound of string
let default_state x =  (* 0, default value? *)
      raise (NotFound “undefined variable”)

(* example of an initial state with x = 1 *)
let x1_state = update "x" (Num "1") default_state

(* test case of a While statement: "skip; x = 5" *)
let test1 = Seq (Skip, Ass ("x", Num "5"))

let new_state = ns (Inter (test1, x1_state))
  in new_state “x”
```