1. (12 pts) OCaml
   a. (2 pts) Give the type of the following OCaml expression
      fun x y -> x (y+2)   Type = (int -> ‘a) -> int -> ‘a
   b. (2 pts) Write an OCaml expression with the following type
      (bool -> int) -> int   Code =  fun x -> 1+(x true)
                              fun x -> [1 ; (x true) ]
                              let f x = 1 + (x true)
                              let f x = [1 ; (x true) ]
   c. (2 pts) Give the value of the following OCaml expression. If an error exists, describe the error.
      (fun x  -> if (x > 0) then x+1 else 1) 1   Value/Error =
      Error = missing else branch = unit (), so type of then branch must match
      Message = this expression has type int but is here used with type unit
   d. (6 pts) Using fold and an anonymous function, write a function attendance which when applied to a list lst of bools, returns the number of elements of lst that are true. Example: attendance [true; false; false; true; true] = 3
      let attendance x = fold (fun a b -> if b then (a+1) else a) 0 x

2. (8 pts) Context free grammars
   a. (2 pts) Write a grammar for $x^y$, where $x = y+3$ (i.e., exactly 3 more $a$’s than $b$’s)
      $S \rightarrow$ aaaL   $L \rightarrow$ aLb  |  epsilon
   b. (6 pts) Consider the following grammar (S = start symbol & terminals = [, ], ;, e):
      $S \rightarrow$ [A]  |  epsilon
      A $\rightarrow$ A ; S  |  e
      i. (3 pts) Present a derivation for the string [e;[e;]]
      $S \Rightarrow [A] \Rightarrow [A;S] \Rightarrow [e;S] \Rightarrow [e;[A]] \Rightarrow [e;[A;S]] \Rightarrow [e;[e;S]] \Rightarrow [e;[e;]]$  leftmost
      $S \Rightarrow [A] \Rightarrow [A;S] \Rightarrow [A;[A]] \Rightarrow [A;[A;S]] \Rightarrow [A;[A;]] \Rightarrow [A;[e;]] \Rightarrow [e;[e;]]$  rightmost
      ...(many other possible derivations)
      ii. (3 pts) Show the parse tree for your derivation

let rec fold f a l = match l with
| [] -> a
| (h::t) -> fold f (f a h) t