1. (15p)
a) x = malloc(10);
   x = malloc(10);
b) Because closures also contain the environment, with the value of the local variables.
c) a"b"c"

2. (10p) (no deductions if the input array is modified)
def map(l)
    l2 = []
    l.each do |x|
        l2.push(yield(x))
    end
    l2
end

3. (15p) (3 points each, no partial credit)
a) bool * int
b) 'a -> 'a
c) int -> int -> bool
d) int * int list -> int list
e) 'a -> ('b -> 'a list) -> 'b -> 'a list

4. (10p)
let rec partition n l =
    match l with
    | [] -> ([],[])
    | (h::t) -> let (l1,l2) = partition n t in
           if h<n then (h::l1, l2)
           else (l1, h::l2)

Another solution using helper functions:

let rec less n l = match l with
    | [] -> []
    | (h::t) -> if h<n then h::(less n t) else (less n t)

let rec greater n l = match l with
    | [] -> []
    | (h::t) -> if h>=n then h::(greater n t) else (greater n t)

let partition n l = (less n l, greater n l)
5. (10p)
let rec testEq f1 f2 l =
  match l with
  [] -> true
  | (h::t) -> if (f1 h) = (f2 h) then (testEq f1 f2 t)
     else false

6. (8p) (-2 if the grammar is not regular, for example:
   S -> aS | Sa | b | \epsilon )
   S -> aS | bT | T
   T -> aT | \epsilon

7. (10p)
   S -> aScc | T
   T -> bTc | \epsilon

8. (12p) (1.5 points each, no partial credit)
   a)
   b) left to right (or just left)
   c) left to right (or just left)
   d) Yes
   e) No
   f) Yes
   g) No
   h) No

9. (5p)
   (a|(b|c)(b|c)(b|c)*)*

10. (5p) (-3 if one or more final state is wrong,
       -3 if one or more transition is wrong)