1. Functional languages (4 points)
   a. Why is Ocaml considered a functional programming language?

   **Focuses on expression evaluation with less emphasis on memory writes;**
   **has higher-order functions; heavy use of recursion, etc.**

   b. Give one reason why someone might want to use a functional language over
      an imperative one. Your answer should *not* be along the lines “…because they
      are better at/like functional languages more…”

   **Easier to parallelize code when there aren’t memory writes**

2. What is a higher order function? That is, what can you do with higher order
   functions that you cannot do with “regular” functions? (2 points)

   **First class variable; can be passed to or returned from a function**
3. Give the types (as Ocaml would) of the following expressions, or ERROR if there is an error: (7 points)

   a. [2;'c'] ________________ error – heterogenous types________
   b. (2,'c') ________________ int * char_________________________
   c. [1;2;3] ________________ int list__________________________
   d. (‘c’;“hello”) __________ error – tuple uses , not ; __________
   e. [[[1;2;2];[2;3]]] __________ int list list________________________
   f. [[[1;2]];[2;3]] __________ error – not same level of nesting_______
   g. Printf.printf “%d” 3 __________ unit__________________________

4. Give the type of the following functions or ERROR if there is an error (4 points):

   a. let head l a = match l with
      | [] -> (1,(1+a)) |
      | _ -> (0,1)
   ;;
   ___’a -> int -> int * int____________________________________

   b. let head l = match l with
      | [] -> [] |
      | _ -> (0,1)
   ;;
   ___error – does not return same type___________________________

5. Give the value of the following Ocaml expressions or ERROR if there is an error (5 points)

   a. let x = 3 in
      x::[] ___________________________ [3]_________________________

   b. let x = 3 in
      x@[1;2;3] __________ error – concat list operator expects 2 lists__
c. \( \text{List.map}((\text{fun } z \rightarrow z + z), [1; 2; 3]) \)  
\[ [2; 4; 6] \]

d. \( \text{let } x = 2 \text{ in} \)  
\( \text{let } y = 3 \text{ in} \)  
\( x::y::[2; 4; 6] \)  
\[ [2; 3; 2; 4; 6] \]

e. \( \text{let list } = [1; 2; 3] \text{ in} \)  
\begin{align*}  
\text{match list with} & \\
[] & \rightarrow -1 \mid \\
(h::t) & \rightarrow \text{if } (h <> 3) \text{ then } 0 \text{ else } 1 
\end{align*}  
\[ 0 \]