CMSC 330 Spring 2016 Quiz #2

Name: ________________________________
Discussion Time: 10am 11am 12pm 1pm 2pm 3pm
TA Name (Circle): Adam Anshul Austin Ayman Damien
Daniel Jason Michael Patrick William

Instructions:

• Do not start this test until you are told to do so!
• You have 15 minutes for this quiz.
• This is a closed book exam. No notes or other aids are allowed.
• For partial credit, show all your work and clearly indicate your answers.
• Write neatly and erase cleanly. Credit cannot be given for illegible answers.
• Code below defines map, fold_left and fold_right functions and is given for reference.

```ocaml
let map f xs = match xs with
  | [] -> []
  | (x::t1) -> (f x)::(map f t1)

let fold_left f a xs = match xs with
  | [] -> a
  | (x::t1) -> fold_left f (f a x) t1

let fold_right f xs a = match xs with
  | [] -> a
  | (x::t1) -> f x (fold_right f t1 a)
```

1. Give the type of following expressions: 2 pts
   
   a) ([1;3;5],4) int list * int
   
   b) fun x y -> x@y 'a list ->'a list ->'a list

2. Give an ocaml expression which matches the following types: 3 pts
   
   a) int -> int -> bool fun a b ->a + b >0
   
   b) int list -> 'a -> 'a

   ```ocaml
   fun lst x -> match lst with
   | [] -> x
   | h::t -> if h > 0 then x else x;;
   ```

   c) ('a -> 'b -> 'c) -> 'b -> 'a -> 'c fun f x y ->f y x
3. **removeAssoc**: Association Lists are a simple map data structure used in OCaml. An association list is a list of tuples, where the first member of the tuple is the key, and the second member of the tuple is the value. Write a function which, given an association list and a value, removes every association for that value. The type for removeAssoc should be \((a \times b) \text{ list} \rightarrow b \rightarrow (a \times b) \text{ list}\). E.g., removeAssoc \([(1, 2); (2, 2); (1, 3)] 2\) evaluates to \([(1, 3)]\). You are not allowed to use for and while loops (0 credit) and there is +1 extra credit for using fold.

```ocaml
let rec remove_assoc l v = match l with |
  | [] -> [] |
  | (key, val)::t -> if val = v then remove_assoc t v |
  | else (key, val)::(remove_assoc (t v)) |

let remove_assoc l v = |
  let rec remove_assoc_helper l v acc = match l with |
  | [] -> acc |
  | (key, val)::t -> if v = val then remove_assoc_helper t v acc |
  | else remove_assoc_helper t v (key, val)::acc |
  in remove_assoc_helper l v []
```


4 pts

4. Write a function isEven using map that takes one argument, a list of ints, and outputs a list of strings: even if the number is even, odd if the number is odd. Remember that 0 is an even number. You must use map and an anonymous function to receive full credit. E.g., isEven \([1;2;3;4]\) evaluates to \["odd","even","odd","even"\].

```ocaml
let is_even l = map (fun x -> if x mod 2 = 0 then "even" else "odd")
```

4pts