CMSC 330, Fall 2017 Quiz 1

Name (as it appears on Gradescope) _______________________________________

Discussion Time (circle one) 10am 11am 12pm 1pm 2pm 3pm

Discussion TA (circle one) Joseph  Greg  Justin  Michael  BT  Daniel  David  Derek  Cameron  Eric  Kesha  Shriraj  Pei-Jo  Michael  Bryan  Kameron

Instructions
● Do not start this quiz until you are told to do so.
● You have 15 minutes for this quiz.
● This is a closed book quiz. No notes or other aids are allowed.
● For partial credit, show all your work and clearly indicate your answers.

1. (2 points each) What is the type of the following OCaml expressions?

   a. `([("hello", 7, true)]
      (string * int * bool) list`

   b. `let foo x y = x :: y :: [1.0]
      float -> float -> float list`

   c. `let foo x y z = (x z) && (y z)
      (\'a -> bool) -> (\'a -> bool) -> \'a -> bool`

2. (2 points each) Write OCaml expressions of the following types without using type annotations.

   a. `string * float list
      ("hello", [3.0])`

   b. `(\'a -> int) -> \'a -> int
      let foo f x = (f x) + 1`
3. **(5 points)** Write an function `cap : float list -> float -> float list` which takes a list of floats `lst` and a float `max`, and returns a list of floats with each float greater than `max` replaced with `max`. You may use `map`, defined below.

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

- `cap [1.0; 4.0; 3.0; 2.0; 5.0] 3.0 = [1.0; 3.0; 3.0; 2.0; 3.0]`
- `cap [1.0; 4.0; 3.0; 2.0; 5.0] 0.0 = [0.0; 0.0; 0.0; 0.0; 0.0]`
- `cap [1.0; 4.0; 3.0; 2.0; 5.0] 6.0 = [1.0; 4.0; 3.0; 2.0; 5.0]`

```ocaml
let cap lst n = map (fun x -> if x +. 0.0 > n then n else x) lst
```

*(perhaps the +. 0.0 should be optional)*

4. **(5 points)** Write a function `range : int -> int -> int list` which takes an int `start` and an int `end` and returns a list of consecutive integers in the range `[start, end)` (excluding `end`).

- `range 0 4 = [0; 1; 2; 3]`
- `range (-2) 2 = [-2; -1; 0; 1]`
- `range 4 4 = []`
- `range 4 2 = []`

```ocaml
let rec range s e =  
  if s < e then  
    s :: (range (s + 1) e)  
  else  
    []
```