CMSC 330, Summer 2018 Quiz 2 (Solution)

Name ____

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. *(6 points)* Give the type of the following OCaml expressions. If there is a type error, explain it.

   (a) `fun x -> 5::x`

   `int list -> int list`

   (b) `if 1 = 1 then true else 0`

   Type Error

   (c) `(let x = 10 in x, let y = fun z -> z in y)`

   `int * ('a -> 'a)`

2. *(6 points)* What do the following OCaml expressions evaluate to? If there is a type error, explain it.

   (a) `(fun x y -> x + y) 1 2`

   `3`

   (b) `(fun x -> x + 1) (fun x -> x * 2) 2`

   Type Error

   (c)
   ```
   let xs = ["a"; "b"; "c"] in
   fold_right (fun x (i, ys) -> (i + 1, (i, x)::ys)) xs (0, [])
   ```

   `(3, [(2, "a"); (1, "b"); (0, "c")])`

3. *(6 points)* Using either `fold_left` or `fold_right` implement the `map` function. You are not permitted to make `map` recursive.

```
(* Returns the resulting list from applying f to each element in xs *)
let map (f : ‘a -> ‘b) (xs : ‘a list) : ‘b list =
  fold_right (fun x a -> (f x)::a) xs []
```

4. *(2 points)* Conversely, provided with only the `map` function could you implement `fold_left` without making it recursive? Explain why or why not.

No. The type of `map` is `(‘a -> ‘b) -> ‘a list -> ‘b list`. This is not sufficiently general to implement `fold_left`.