CMSC 330 Quiz 2 Spring 2022

Q1. OCaml Typing

- For all following three sub-questions, you are **not allowed to use type annotations**.
- All pattern matching **must be exhaustive**.
- No other warnings should be raised.

Q1.1. Give an expression of the following type: `int -> int -> int * bool * int`

```
fun x y -> (x+1, true, y-1)
```

Q1.2. Give an expression of the following type: `'a -> 'a -> bool`

```
fun x y -> x = y
```

Q1.3. Give an expression of the following type: `('a -> 'b) -> 'a -> 'b -> bool`

```
fun f x y -> (f x) = y
```

Q2. OCaml Typing

The following OCaml expression does not type check.

```
if 1 then if true then 1 else 2.0 else 3.0
```

Identify and state the type error(s).

1. Condition for `if` must have type `bool`   2. Mismatched return types (either all `int` or `float`)

Fix and rewrite the given OCaml expression such that it type checks.

```
if true then if true then 1.0 else 2.0 else 3.0
```

Q3. OCaml Coding with Recursion

Implement a function `swap` that swaps the position of elements in a list pairwise. In the case of a list with an even number of elements, this is pretty straightforward. However, if the list has an odd number of elements, the position of the "extra" element should be unchanged.

Notes: Your pattern matching must be exhaustive for full credit. You are not allowed to use the `List` module.

Examples:

```
swap [] = []
swap [1] = [1]
swap [1; 2] = [2; 1]
swap [1; 3; 3; 7] = [3; 1; 7; 3]
swap [1; 2; 3; 4; 5] = [2; 1; 4; 3; 5]
```

```
let rec swap lst =
  match lst with
  | [] -> []
  | h::h'::t -> h'::h::(swap t)
  | h::t -> h::(swap t)
```

Q4. OCaml Fill in the Blanks

Given the following `fold_right` implementation, implement a function called `divisible_by_7` which returns a tuple whose first value is the sum of all elements in the list which are divisible by 7 and the second value is a list containing those elements **in order**.

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

Examples:

```
divisible_by_7 [] = (0, [])
divisible_by_7 [1; 5; 7; 3; 14; 21; 0; 8] = (42, [7; 14; 21; 0])
```

Note: You are not allowed to use the `List` module.

Prompt:

```
let divisible_by_7 lst =
  fold_right (fun x (s, l) -> if x mod 7 = 0 then (s + x, x::l) else (s, l)) lst (0, [])
```