COL 765: Introduction to Logic and Functional Programming
Quiz 4, 29.08.2024
(Programming with lists)

Name: _____ Entry No. _____

Consider the following data type definition for representing vectors as lists:

```
type 'a vector = 'a list;;
```

Q1 [5] Write a function

```
zip: 'a vector -> 'b vector -> ('a * 'b) vector
```

(a program in OCaml) which given vector `v1 = [` x_1; … ; x_n `]` of type `'a vector` and `v2 = [` y_1; … ; y_n `]` of type `'b vector`, presumably both of the same length (dimension), returns a vector of the same length, the i^{th} entry of which is the pair (x_i, y_i). What should you do if the lengths (dimensions) of `v1` and `v2` are unequal?

```
exception UnequalLength;;
let rec zip v1 v2 = match v1, v2 with
      [ ], [ ] -> [ ]
    | [ ], _ -> raise UnequalLength
    | _, [ ] -> raise UnequalLength
    | x::xs, y::ys -> (x,y):: (zip xs ys)
;;
```

Q2 [5] Recall that the "dot product" of two vectors of equal length `v1 = [` x_1; … ; x_n `]` and `v2 = [` y_1; … ; y_n `]` is defined as $\Sigma_{i=1}^{n}\ x_i * y_i$.
Write a program `dotprod: float vector -> float vector -> float` using `zip`, `map`, and `fold_left` – with appropriate function arguments for `map` and `fold_left`, and an initial value for `fold_left`.
(No marks will be given unless you use these standard list functions). You may find it useful to use the **`let`** `__` **`in`** `___` construct to name the argument functions used in `map` and `fold_left`, and for the intermediate results.

```
let dotprod v1 v2  =
     let mult (x, y) = x*y
     and sum x y = x+y
     and paired = zip v1 v2
       in fold_left sum 0 (map mult paired)
;;
```