

NAME \_\_\_\_\_

TEACHING ASSISTANT

Barath Cameron Olasubomi Vincent Pavan Michael Cliff Shruti Salma Shilpa  
Mihailo Arun Danielle Noah Chris Liam Suteerth Sabrina

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 pts] Give the type of the following OCaml expression. If there is a type error, explain why the expression would result in a type error.

(a) `fun x -> x + 3`

(b) `[] :: [] :: []`

(c) `fun x y z -> if x y > x z then (x y) else (z *. 5.0)`

2. [6 pts] Give an OCaml expression of the following type without using type annotations.

(a) `int -> float -> float`

(b) `(int -> int -> int) -> float -> int`

For the below question, you may use the following functions.

```
let rec map f l =
  match l with
  | [] -> []
  | h :: t -> (f h) :: (map f t)

let rec foldl f acc l =
  match l with
  | [] -> acc
  | h :: t -> foldl f (f acc h) t

let rec foldr f l acc =
  match l with
  | [] -> acc
  | h::t -> f h (foldr f t acc)
```

3. [8 pts] Write a function `check_matrix` which applied to `lst`, an argument of type `'a list list`, returns whether `lst` is a well-formed matrix, meaning that the number of elements in each sub-list is the same. Note that the matrix does not have to be a "square matrix," so the number of rows and columns do not have to be equal. `check_matrix` should return true if `lst` is empty.

You **may not** define the following function as recursive. You also **may not** define a recursive helper function, but you can define as many non-recursive functions as you would like.

```
check_matrix [[1 ; 2]; [2; 4]; [3; 6]] = true
check_matrix [[1 ; 2; 3]; [2; 4]; [3; 6]] = false

let check_matrix (lst : 'a list list) : (bool) =
```