

15-150: Principles of Functional Programming

Quiz 4 Practice

Summer 2022

Name: David M Kahn

Andrew ID: davidkahn

Circle the lab you are in:

A	Julia and Yosef	GHC 5207
B	Juhi, Nicole, and Susan	GHC 5208
C	Nancy and Shengchao	GHC 5210
D	Thea and Cindy	GHC 4303
E	Jacky, Jonathan, and Sonya	WEH 7500
F	David	GHC 6503

Question	Points	Score
Practice	0	0
Total:	0	0

Answer the questions in the spaces provided. You will have the full 80 minutes of class time to complete this quiz. Good luck!! Y'all got this.

Question 1: Practice

Consider the following code:

```
fun exists p lst =
  (case lst of
    [] => false
  | x::xs => p x orelse exists p xs
  )
```

- (a) (0 points) Give an extensionally equivalent implementation of `exists` using `foldl` as the only form of recursion.

$$\text{fun exists } p = \text{foldl } (\text{fn } (x, \text{acc}) \Rightarrow \text{acc orelse } p\ x) \text{ false}$$

$\uparrow \qquad \qquad \qquad \uparrow$

note: order critical if p not total

- (b) (0 points) Give an implementation of `existsCPS` satisfying $\text{existsCPS } p \text{ lst } k \cong k \text{ (exists } p \text{ lst)}$ for any values $p : 'a \rightarrow \text{bool}$, $\text{lst} : 'a \text{ list}$, and $k : \text{bool} \rightarrow 'b$.

$$\text{fun existsCPS } p \text{ lst } k =$$

$$\text{ (case lst of}$$

$$\text{ [] } \Rightarrow k \text{ false}$$

$$\text{ | x::xs } \Rightarrow$$

$$\text{ if } p\ x$$

$$\text{ then } k \text{ true}$$

$$\text{ else existsCPS } p \text{ xs } k$$

note:
careful of
nontotal p !

Consider the following code snippet:

```
fun existsExn p lst =  
  (case lst of x::xs =>  
    if p x  
    then true  
    else existsExn p xs  
  )  
  handle EXN => false
```

- (c) (0 points) Which exception should replace `EXN` so that `existsExn p` $\cong$ `exists p` for total `p`?

(c) Match

- (d) (0 points) David decides to try to be clever and replace `EXN` in the above code with the wildcard pattern `_` so that he doesn't need to think about what the answer to the previous question should be. Does his solution work? Why or why not?

It does work. If `p` is total, no other exceptions can be raised.

Consider the following code:

```
fun foo f lst1 lst2 = map (fn x => x + f lst1) lst2  
  
fun bar f lst1 =  
  let  
    val fl1 = f lst1  
  in  
    fn lst2 => map (fn x => x + fl1) lst2  
  end
```

- (e) (0 points) Select all of the following that are extensionally equivalent:

- ☐ `foo` and `bar`
- ☐ `foo f` and `bar f`
- ☐ `foo f lst1` and `bar f lst1`
- ☐ `foo f lst1 lst2` and `bar f lst1 lst2`

none - if `f lst1` not valuable
compare when `lst2 = []`

- (f) (0 points) Select all of the following that are extensionally equivalent, assuming `f` total:

- ☒ `foo` and `bar` *
- ☒ `foo f` and `bar f`
- ☒ `foo f lst1` and `bar f lst1`
- ☒ `foo f lst1 lst2` and `bar f lst1 lst2`

* slight ambiguity - I mean
`foo` and `bar` only considering
total first argument

note: all options in each of f & e
are equivalent by definition

Consider the following code snippet:

```
datatype 'a tree = Leaf | Node of 'a tree * 'a * 'a tree

fun depth t =
  (case t of
    Leaf => 0
  | Node (t1, x, t2) => 1 + Int.max (depth t1, depth t2)
  )

fun depthCPS t k =
  (case t of
    Leaf => CASE1
  | Node (t1, x, t2) => CASE2
  )
```

- (g) (0 points) Give a code snippet to replace `CASE1` so that `depthCPS` is the continuation passing style version of `depth` for an arbitrary continuation `k`.

$k \circ$

- (h) (0 points) Which of the following code snippets should replace `CASE2` so that `depthCPS` is the continuation passing style version of `depth` for an arbitrary continuation `k`?

- A. `1 + Int.max (depthCPS t1 (fn y => y), depthCPS t2 (fn z => z))`
- B. `1 + Int.max (depthCPS t1 k, depthCPS t2 k)`
- C. `k (1 + Int.max (depthCPS t1 k, depthCPS t2 k))`
- D. `Int.max (depthCPS t1 (fn y => k (y + 1)), depthCPS t2 (fn z => k (z + 1)))`
- E. `k (Int.max (depthCPS t1 (fn y => k (y + 1)), depthCPS t2 (fn z => k (z + 1))))`
- ☒ F. `depthCPS t1 (fn y => depthCPS t2 (fn z => k (1 + Int.max (y,z))))`
- G. `k (depthCPS t1 (fn y => depthCPS t2 (fn z => k (1 + Int.max (y,z))))`

- (i) (0 points) Give the most general type of `foldl (fn (x, y) => raise x)`, or state that none exists.

$'a \rightarrow \text{exn} \rightarrow \text{list} \rightarrow 'a$

- (j) (0 points) Give the most general type of `map map`, or state that none exists.

$('a \rightarrow 'b) \text{list} \rightarrow ('a \text{list} \rightarrow 'b \text{list}) \text{list}$