15-150: Principles of Functional Programming

Quiz 4 Practice

Summer 2022

Name: _____

Andrew ID: _____

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

Question	Points	Score
Practice	0	
Total:	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.

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

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) _____

(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?

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`

(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`

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`.

(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.

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