15-150: Principles of Functional Programming

Quiz 2 Practice

Summer 2022

Name: David M Kahn

Andrew ID: david kah

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	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 function:

```
(* ofact : int -> int option
 * REQUIRES: true
 * ENSURES: ofact n ==> SOME (n!) if n >= 0, otherwise NONE
 *)
fun ofact (n : int) : int option =
 (case (n < 0, n) of
       (true, _) => NONE
     | (_, 0) => SOME 1
     | _ => (case ofact (n-1) of
                  NONE => NONE
                | SOME r => SOME (n * r)
             )
 )
```

(a) (0 points) **True** or **False**. Is `ofact` tail recursive.

(b) (0 points) `ofact` is...

- ☑ a value
- ☑ valuable
- ☑ total
- ☐ not well-typed

(c) (0 points) The tighest big-O bound for `ofact` is...

A. $O(n!)$

B. $O(n^2)$

C. $O(n)$

D. $O(\log n)$

E. $O(1)$

(d) (0 points) Suppose we are trying to prove the theorem:

For all values `n : int` where $n \geq 0$, `ofact n` $\cong$ `SOME (fact n)`.

Fill in the proof outline below. You do not need to prove the BC and IS, just state what the case would be. Make sure to clearly restate the theorem in the IH (don't just say "the theorem holds").

We proceed by numerical induction on n. (or simple or weak induction)

BC: Want to prove the theorem for n=0

(proof omitted)

IH: For some $k \geq 0$ ofact k ≅ SOME (fact k)

IS: Want to prove the theorem for n = k+1

(proof omitted)

(e) (0 points) Implement the following function using tail recursion. You may write as many helper functions as needed:

```
(* dotProd : int list * int list -> int
 * REQUIRES: length L1 = length L2
 * ENSURES: dotProd ([x0, x1, ..., xn], [y0, y1, ..., yn])
 *               ==> x0 * y0 + x1 * y1 + ... + xn * yn
 *)
```

```
fun dotHelp (lsts : int list * int list, acc: int) : int =
  (case lsts of
      ([],[]) => acc
    | (x::xs, y::ys) => dotHelp ((xs, ys), x*y + acc)
  )

fun dotProd (lsts: int list * int list) : int =
    dotHelp (lsts, 0)
```