COMP 212 : Functional Programming, Spring 2023

Homework 08

Name: ____________________

Wes Email: ____________________

Question	Points	Score
1	12	
2	20	
Total:	32	

If possible, please type/write your answers on this sheet and upload a copy of the PDF to your google drive handin folder. Otherwise, please write the answers in some sort of word processor and upload a PDF. Please name the file `hw08-written.pdf`.

1. **Analysis**

   (a) The following append function was a task in lab (see the lab handout and the lecture notes for Lect 15-16 for an explanation of how tabulate works):

```
fun myAppend (s1 : 'a Seq.seq, s2 : 'a Seq.seq) : 'a Seq.seq =
    Seq.tabulate (fn i => case i < Seq.length s1 of
                     true => Seq.nth (i, s1)
                   | false => Seq.nth (i - (Seq.length s1), s2),
                  Seq.length s1 + Seq.length s2)
```

(2) i. Give a tight O-bound for the work of `myAppend`. Make sure you explicitly state what quantities you are analyzing the work in terms of. Briefly explain why your answer is correct.

Solution:

(2) ii. Give a tight O-bound for the span of `myAppend`. Make sure you explicitly state what quantities you are analyzing the span in terms of. Briefly explain why your answer is correct.

Solution:

(b) Consider the following reverse function:

```
fun reverse' (s : 'a Seq.seq) : 'a Seq.seq =
    Seq.reduce (fn (x,y) => myAppend (y, x),
                Seq.empty(),
                Seq.map (Seq.singleton, s))
```

`Seq.singleton` and `Seq.empty` take constant time.

(2) i. Give a tight O-bound for the work of `reverse'`, in terms of the length of `s`. Briefly explain your answer.

Solution:

(2) ii. Give a tight O-bound for the span of `reverse'`, in terms of the length of `s`. Briefly explain your answer.

Solution:

(c) Consider the following alternative implementation of the reverse function:

```
fun reverse (s : 'a Seq.seq) : 'a Seq.seq =
    Seq.tabulate (fn i => Seq.nth ((Seq.length s) - (i + 1), s), Seq.length s)
```

(2) i. Give a tight O-bound for the work of `reverse`, in terms of the length of `s`. Briefly explain why there is a discrepancy between this and the work of `reverse'`.

Solution:

(2) ii. Give a tight O-bound for the span of `reverse`, in terms of the length of `s`. Briefly explain why there is a discrepancy between this and the span of `reverse'`.

Solution:

2. **NON-COLLABORATIVE CHALLENGE PROBLEM: Tree Shrinking**

Remember that non-collaborative challenge problems are to be done independently. You are not allowed to communicate with anyone about the problems, except to ask the instructor or TAs clarification questions (not hints). Additionally, you are not allowed to search for help on the specific problem from any sources besides the course materials.

Sometimes, a computation will produce a tree with patterns like `Node(Empty,t)` or `Node(t,Empty)` in it, which can be optimized to just `t` without changing the contents of the tree. The following function does this:

```
fun shrink (t : 'a tree) : 'a tree =
    case t of
        Empty => Empty
      | Leaf x => Leaf x
      | Node (l,r) => (case (shrink l, shrink r) of
                            (Empty, r') => r'
                          | (l',Empty) => l'
                          | (l',r') => Node(l',r'))
```

That is, if either subtree of a tree shrinks to the empty tree, we delete that node, and otherwise we make a node of the shrunken subtrees.

In homework 7, you implemented `reduce` for trees.

```
fun reduce (n : 'a * 'a -> 'a, e : 'a, t : 'a tree) : 'a = ...
```

In this problem, you will prove that the behavior of `reduce` is unchanged by shrinking:

Theorem 1. *Suppose we have total `n:'a * 'a -> 'a` and `e:'a` such that*

1. *For all `x`, `n(e,x)` $\cong$ `x`*
2. *For all `x`, `n(x,e)` $\cong$ `x`*

Then for all `t`, `reduce(n, e,, shrink t)` $\cong$ `reduce(n, e, t)`

(20) (a) Prove this theorem by induction on t.

Solution:

Solution:

Solution: