COMP 212 : Functional Programming, Fall 2024

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 last week 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. To analyze the running time of `Seq.reduce`, you can assume it is implemented like your tree implementation from HW07, run on a balanced tree; use a recurrence.

(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 PROBLEM: Tree Proof**

Remember that non-collaborative 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.

In this problem, you will prove a specification about the `filter_less` and the `depth` functions on trees from Homework 5. To avoid confusion, remember that these examples used the following tree type with data at the internal nodes:

```
datatype tree = Empty | Node of tree * int * tree
```

The code for these functions is

```
fun depth (t : tree) : int =
    case t
     of Empty => 0
      | Node(l,_,r) => 1 + Int.max(depth l, depth r)

fun combine (t1 : tree, t2 : tree) : tree =
    case t1 of
        Empty => t2
      | Node(l1,x1,r1) => Node(combine(l1,r1),x1,t2)

fun filter_less(t : tree, i : int) : tree = ... your HW5 solution ...
```

For the running time analysis of `quicksort` on trees, we need the following property:

Theorem 1. *For all trees `t:tree` and `i:int`,*

$$\mathit{depth}(\mathit{filter\_less}(t,i)) \leq \mathit{depth}(t)$$

(20) (a) Prove this theorem for your `filter_less` from homework 5.[1] Prove any lemmas about `combine` that you need. You can use the properties of `max` (maximum) from Homework 5 without proving them.

[1] If you didn't get full credit and can't infer a corrected solution from the comments on your submission, please contact me for the solution.

Solution:

Solution:

Solution:

Solution: