CS312 Fall 2000: Preliminary Exam II

November 14, 2000

There are 3 pages and 14 questions in this exam. Please take a minute to check that you have all of the pages. You have 1.5 hours to complete the exam. Write all of the answers to the questions in the exam booklets provided. Make sure to put your name on each exam booklet that you use and turn them all in. Good luck!

Throughout the exam, you will be working with the SML definitions given below:

```
datatype 'a tree =
  Leaf | Node of {datum:'a, left:'a tree, right:'a tree}

val x = Node{datum=1, left=Leaf, right=Leaf}
val y = Node{datum=3, left=Leaf, right=Leaf}
val z = Node{datum=2, left=x,    right=y}

fun height(t:'a tree):int =
    case t of
      Leaf => 0
    | Node{datum,left,right} =>
        1 + Int.max(height left, height right)

fun tfold (f:('a*'b*'a) -> 'a) (accum:'a) (t: 'b tree): 'a =
    case t of
      Leaf => accum
    | Node{datum,left,right} =>
        f(tfold f accum left, datum, tfold f accum right)

fun f(t:'a tree):'a tree =
    let fun g(l:'a tree, d:'a, r:'a tree):'a tree =
              Node{left=r, datum=d, right=l}
    in
        tfold g Leaf t
    end

val h : int -> int tree -> bool =
    fn (i:int) => let fun j(lf:bool, d:int, rt:bool):bool =
                            lf orelse rt orelse (d = i)
                  in
                      tfold j false
                  end
```

1 [5 points]. Suppose we have a tree of height *n*. What is the maximum number of nodes in

the tree?

A: 2^(n+1) – 1 this is a good one to prove by induction...

2 [5 points]. Using `tfold`, write a function `tsum` that sums all of the data in a tree of integers. Your function should have the signature:

```
val tsum : int tree -> int
```

and be as simple as possible

A: `val tsum = tfold (fn (x,y,z) => x+y+z) 0`

3 [7 points]. Using `tfold`, write a function `tflatten` that returns all of the data in a tree in a list. The left-most datum should appear first in the list, and the right-most datum should appear last. Your function should be as simple as possible and have signature:

```
val tflatten : 'a tree -> 'a list
```

A: `val tflatten = tfold (fn (x,y,z) => x@(y::z)) []`

4 [2 points]. To what value does `f(Leaf)` evaluate?

A: `Leaf`

5 [4 points]. To what value does `f(z)` evaluate?

A: `Node{datum=2, left=Node{datum=3,left=Leaf,right=Leaf}, right={datum=1,left=Leaf,right=Leaf}}`

6 [15 points]. Using induction, prove that for all trees, `f(f(t)) = t`. To receive full credit, be sure to state carefully the property you are trying to prove, your induction hypothesis, your base case, your inductive case, and where you use the induction hypothesis.

A: We want to prove that for all trees t, P(t) is true where P(t) is "f(f(t)) = t". We will prove this by strong induction on the height of the tree.

Base case: height=0. Then t=Leaf. Thus f(Leaf) = Leaf. So f(f(Leaf) = Leaf.

Inductive case: Assume that for all i <= n, if height(t) = i, then P(t) is true. We must show that for a tree t of height n+1, P(t) is true. Since the height of t is n+1, it must be of the form Node {datum=d,left=l,right=r} where the heights of the left and right sub-trees are at most n. Then f (t) = f(Node{datum=d,left=l,right=r}) = Node{datum=d,left=f(r), right=f(l)}. Thus, f(f(t)) = f(Node {datum=d,left=f(r),right=f(l)}) = Node{datum=d,left=f(f(l)), right=f(f(r))}. Now by the induction

hypothesis, since l and r are trees of strictly smaller height than t, f(f(l)) = l and f(f(r)) = r. Thus, f(f(t)) = Node{datum=d,left=l,right=r} = t.

7 [4 points]. To what value does `(h(2))(z)` evaluate?

A: true

8 [4 points]. To what value does `(h(4))(z)` evaluate?

A: false

9 [10 points]. Using big-O notation, what is the worst-case running time of `(h(i))(t)` for a tree `t` with *n* nodes?

A: O(n) – at worst, each node in the tree will be visited once.

10 [15 points]. An *ordered* tree is one with the property that for any node, the datum is greater than or equal to the data in the left sub-tree, and less than the data in the right sub-tree. A *balanced* tree is one a tree with the property that for any node, the number of nodes in the left and right sub-trees are essentially the same.

Write a new function `h2` that, when given an integer `i` and an ordered, balanced tree `t`, returns the same result as `h`, but is asymptotically faster.

A:
```
fun h2 (i:int) (t:tree) =
    case t of
      Leaf => false
    | Node{datum,left,right} =>
      (case Int.compare(i,datum) of
         Equal => true
       | Less => h2 left
       | Greater => h2 right)
```

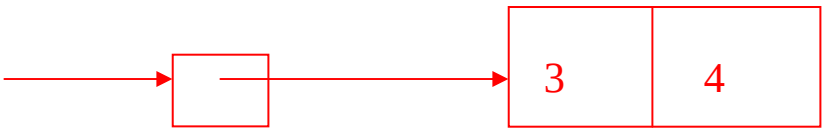
11 [10 points]. Using big-O notation, what is the worst-case running time of your `(h2(i))(t)` for a tree `t` with *n* nodes?

A: O(lg n)

12 [5 points]. For the next few problems, you will be using box-and-pointer diagrams as drawn in class, section, and the notes, to indicate the result of memory after evaluating an expression. Please keep your drawings as neat as possible.

What does memory look like after evaluating:

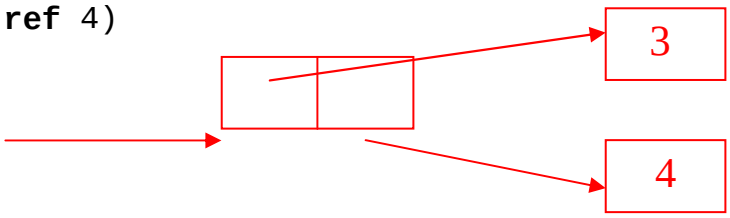
```
ref (3,4)
```



13 [5 points]. What does memory look like after evaluating:

```
(ref 3, ref 4)
```

?



14 [10 points]. What does memory look like after evaluating:

```
let val x = (3,4)
```

```
    val y = (x,x)
    val f = (fn x => fn (z => x + z + (#1 y))
in
    f(#2 x)
end
```