Computer Science 312

Fall 2006

Prelim 1
March 9, 2006

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grade</td>
<td>/12</td>
<td>/10</td>
<td>/10</td>
<td>/14</td>
<td>/8</td>
<td>/18</td>
<td>/14</td>
<td>/14</td>
<td>/100</td>
</tr>
<tr>
<td>Grader</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. (12 points) Given the following data type definition for a binary tree

```ml
datatype 'a tree =
    Node of 'a tree * 'a * 'a tree | Empty
```

Consider the following in-order fold function from Problem Set 2 that does recursive folds on the left and right subtrees,

```ml
fun fold (f:'b*'a*'b->'b) (b:'b) (t:'a tree):'b =
    case t of
        Empty => b
    | Node(l,v,r) => f(fold f b l,v,fold f b r)
```

Use this definition of fold to complete the definition of the function `countLeaves` so that it counts the number of leaf nodes in the tree (where a leaf node is one that has two `Empty` sub-trees).

For example, given the tree,

```ml
val oneleaf =
    Node(Empty,2,Node(Empty,3,Node(Empty,4,Empty)))
```

`countLeaves(oneleaf)` is 1. Similarly, `countLeaves(Empty)` is 0.

You will receive 10 of 14 points on this problem for an answer that produces the correct result and uses the `fold` function above. You will receive full credit if your answer further makes no use of conditionals (case statements, if-then-else, pattern matching, etc.).

```ml
fun countLeaves (t: 'a tree) =
    fold (fn(l,v,r) => (l+r) div 2 * 2) 1 t div 2
```

Or using a conditional

```ml
fold (fn(l,v,r) => if l=0 andalso r=0 then 1 else l+r) 0 t
```
2. (10 Points) Recall that \texttt{foldl}, \texttt{foldr} and \texttt{map} on lists can be defined as:

\begin{verbatim}
fun foldl f a [] = a |
  foldl f a (h::t) = foldl f (f (h, a)) t

fun foldr f a [] = a |
  foldr f a (h::t) = f(h, (foldr f a t))

fun map f [] = [] |
  map f (h::t) = f(h) :: map f t
\end{verbatim}

Complete the following definition of \texttt{map} that produces the same results as the one above, however is implemented using \texttt{foldl} or \texttt{foldr} (Hint: only one of these will produce a correctly ordered result list).

\begin{verbatim}
fun map f s =
  foldr (fn (x, y) => (f x) :: y) [] s
\end{verbatim}

3. (10 points) Recall that the Fibonacci numbers are defined such that

\begin{verbatim}
fib(n) = fib(n-1) + fib(n-2)
\end{verbatim}

where \texttt{fib(1)=fib(2)=1}.

A naïve implementation of this has a running time that is exponential in \(n\). Complete the following implementation of \texttt{fib} such that it runs in time \(O(n)\).

\begin{verbatim}
fun fib (n) =
  let fun helper(a,b,n) =
    if n<=1 then b
    else helper(b, a+b, n-1)
  in
    helper(0,1,n)
  end
\end{verbatim}
4. (14 Points) Consider the following function

val empty = fn _ => raise Fail "Empty."

And two functions insert and lookup that have the following behavior. After inserting (a,b) a lookup of a returns b and a lookup of b returns a. For example:

- val onetwo = insert (insert empty (1,10)) (2,20);  
  val onetwo = fn : int -> int
  - lookup onetwo 1;
  val it = 10 : int
  - lookup onetwo 10;
  val it = 1 : int
  - lookup onetwo 2;
  val it = 20 : int
  - lookup onetwo 20;
  val it = 2 : int
  - lookup onetwo 3;
  uncaught exception Fail: Empty.

Complete the following definitions for the functions insert and lookup (recall that 'a is a parameterized type that must support equality.

fun insert (m:'a->'a) (k:'a,v:'a) (x:'a) =  
  if x=k then v else if x=v then k else (m x)

fun lookup (m:'a->'a) (k:'a) =  
  (m k)
5. (8 Points) Given the following two function definitions

```ml
fun trip f x = f(f(f x))
fun inc x = x+1
```

What is the value of the following expression?

```ml
trip trip inc 12
```

39

6. (18 Points) Which of the signatures A-H listed below correspond to the function \( f \) in each of the following expressions?

```ml
val f = fn _ => print "foo"
```

G) \( \text{val } f = \text{fn} : 'a \rightarrow \text{unit} \)

```ml
fun f g x = g(x)
```

D) \( \text{val } f = \text{fn} : ('a \rightarrow 'b) \rightarrow 'a \rightarrow 'b \)

```ml
fun f x y = let val x = y in (x;y) end
```

A) \( \text{val } f = \text{fn} : 'a \rightarrow 'b \rightarrow 'b \)

B) \( \text{val } f = \text{fn} : ('a \rightarrow 'b) \rightarrow 'b \)

C) \( \text{val } f = \text{fn} : ('a \rightarrow 'b) \times 'a \rightarrow 'b \)

D) \( \text{val } f = \text{fn} : ('a \rightarrow 'b) \rightarrow 'a \rightarrow 'b \)

E) \( \text{val } f = \text{fn} : 'a \times 'b \rightarrow 'a \rightarrow 'b \)

F) \( \text{val } f = \text{fn} : \text{unit} \rightarrow 'a \)

G) \( \text{val } f = \text{fn} : 'a \rightarrow \text{unit} \)

H) \( \text{val } f = \text{fn} : \text{unit} \rightarrow \text{unit} \)
7. (14 Points) In this problem you are to use the pattern matching style of functions (as in the definitions of fold and map in problem 2). *You may not use any form of conditional* (case, if-then-else, etc.).

Given two Boolean arguments nand determines whether either of them is false:

- nand(true, true);
  val it = false : bool
- nand(true,false);
  val it = true : bool
- nand(false,true);
  val it = true : bool
- nand(false,false);
  val it = true : bool

Write a pattern matching function for nand that has as few patterns as possible.

fun nand(true, true) = false
| nand(_,_) = true

Given a natural number and a list firstn returns the first n elements of the list:

- firstn(3,[1,2,3,4,5,6])
  val it = [1,2,3] : int list

Write a pattern matching function for firstn that has as few patterns as possible.

fun firstn(0, _) = []
| firstn(_,[]) = []
| firstn(n, h::t) = h::firstn(n-1,t)
8. (14 Points) Consider the following signature and partial implementation of an ordered set that has operations to create a singleton set, to compute the union of two sets and to remove an element from a set. Note that union and remove take as one of their arguments a comparator function, which given two elements of type 'a determines whether the first argument is less than, greater than or equal to the second.

```plaintext
signature ORDEREDSET =
  sig
    type 'a oset
    val singleton: 'a -> 'a oset
    val union: 'a oset * 'a oset * ('a * 'a -> order) -> 'a oset
    val remove: 'a oset * 'a * ('a * 'a -> order) -> 'a oset
  end

structure oset : ORDEREDSET =
  struct
    type 'a oset = 'a list
    fun singleton(a) = [a]
    fun union(s,t,cmp) = ?????
    fun remove(s,a,cmp) =
      case s of ([]) => []
      | h::t =>
        case cmp(h,a) of
          EQUAL => t
          LESS => h::remove(t,a,cmp)
          GREATER => s
    end

What missing code should go in place of ??? to complete this implementation?

```