Midterm Examination: Sample Solutions

February 23, 1995

Binding and Scope [20 points]

1. `44`. The second binding of `x` does not affect the values of `a` or `b`.
2. `8`. The binding of `x` in the `let` does not affect the value of `a`.
3. `8`. The inner binding of `y` shadows the first one, and does not affect the binding of `b`.

1. `21`. The use of `f` in the body of the second definition of `f` refers to the first binding, not itself.
2. `8`. The use of `f` in the body is a recursive call to the function itself: we're computing the `5`th Fibonacci number.
3. `74`. The re-binding of `y` has no affect on the binding of `f`. To evaluate the binding of `f` we first substitute the value of `y` (which is `7`), then evaluate to get the binding `fn x:int=>x*x+7*7` for `f`.

Induction and Recursion [30 points]

The definition of `segment` is as follows:

```
(* segment (l,i,j) ->* l[i:j], where 0<=i<=j<n=length l *)
fun segment (h::_,0,0) = [h]
  | segment (h::t,0,j) = h :: segment (t,0,j-1)
  | segment (_::t,i,j) = segment (t,i-1,j-1)
```

1. The proof is by induction on `j`.
2. Basis: `j`=`0`. Since `0`≤`i`≤`j`, it follows that `i`=`0`. Since `l`=`h::t` for some `h` and `t`, we have that `l[0:0]`=`[h]`.
3. Inductive hypothesis: assume `j`>`0` and `segment(l,i,j)`$\rightarrow^*$ `l[i:j]`.
4. We are to show that `segment(l,i,j+1)`$\rightarrow^*$ `l[i:j+1]`. We have two cases to consider, according to whether `i`=`0` or `i`≠`0`.
   (a) If `i`=`0`, then

       `segment(h::t,i,j+1)`$\rightarrow^*$ `h::segment(t,0,j+1-1)`.

   By the inductive hypothesis

       `segment(t,0,j)`$\rightarrow^*$ `t[0:j]`.

   But `(h::t)[0:j+1]`=`h::t[0:j]`.

   (b) If `i`≠`0`, then

       `segment(h::t,i,j+1)`$\rightarrow^*$ `segment(t,i-1,j+1-1)`.

   By the inductive hypothesis

       `segment(t,i-1,j)`$\rightarrow^*$ `t[i-1:j]`.

   But `(h::t)[i:j+1]`=`t[i-1:j]`.

Abstract Data Types: Polynomials [20 points]

1.

$$
\begin{aligned}
(a + p\,x) + (b + q\,x) &= (a + b) + (p + q)\,x \\
(a + p\,x) \cdot q &= a\,q + (p\,q)\,x
\end{aligned}
$$

2.

```
fun add_poly (nil,q:real list) = q
  | add_poly (p,nil) = p
  | add_poly (a::p,b::q) = (a+b) :: add_poly(p,q);
```

3.

```
fun mul_poly (nil,q) = nil
  | mul_poly(a::p,q) = add_poly(smul_poly(a,q),0.0::mul_poly(p,q))
```

where

```
fun smul_poly(a:real,nil)=nil
  | smul_poly (a,b::q) = (a*b)::smul_poly(a,q);
```

4. The polynomial in question is $(x - a_0)(x - a_1) \cdots (x - a_{n-1})$. This leads to the following definition:

```
fun poly_with_roots [a] = [0.0,~a]
  | poly_with_roots (a::rs) = mul_poly ([~a,1.0], poly_with_roots rs);
```

Recursion [30 points]

```
fun max (m:int,n:int) = if m<n then n else m;
val max_list = fn l => fold max l 0;
type bid = { start:int, finish:int, rent:int };
fun max_rent (day:int, bids:bid list) =
    let
        fun rent_list (nil:bid list) = nil
          | rent_list ({start=s,finish=f,rent=r}::bl) =
            if f <= day then
                (max_rent (s-1, bids) + r) :: (rent_list bl)
            else
                rent_list bl
    in
        max_list (rent_list bids)
    end;
(* Extra credit *)
fun max_rent (day, bids) =
    let
        fun mr (0, _) = (0, nil)
          | mr (_, nil) = (0, nil)
          | mr (day, (b as {start=s,finish=f,rent=r}) :: bl) =
            if f <= day then
                let
                    val (r_wo, bl_wo) = mr (day, bl)
                    val (r_w, bl_w) = max_rent (s-1, bids)
                in
                    if r_w + r >= r_wo then
                        (r_w + r, b :: bl_w)
                    else
                        (r_wo, bl_wo)
                end
            else
                mr (day, bl)
    in
        mr (day, bids)
    end ;
max_rent(10,[{start=1,finish=3,rent=4},
             {start=2,finish=6,rent=4},
             {start=5,finish=7,rent=4},
             {start=8,finish=10,rent=4}]);
```