1-

a) val a=fn: int*int→ int, since (+) and (*) and (2) are integers
b) val b = fn : real * real → real , since 2.0 is real
c) val c= fn : ('a → 'b) → ('a → 'b) , since it is a higher order function and f: ('a → 'b)
d) val d: (('a → 'a) * 'a) → 'a , for inner F we have F: ('a → 'b) and for outer F we have F: ('b → 'c) . because of unification we are inferring that 'a = 'b = 'c and also F : ('a → 'a)

e) val e = fn: 'a * 'a * ('a → bool) → 'a , since x and y must have the same type and output of function (b) must be Boolean .

2-

val sort= fn: ('a * 'a →bool) * 'a list →'a list

val insert= fn: ('a * 'a →bool) * 'a * 'a list →'a list

as we know, less function has the type 'a * 'a →bool and we can infer sort and insert's types based on their inputs and outputs easily.

4-

if we assume Y as the fixed point combinator we have this type because as we know, concepts of fix point and functional are valid when our domain and range is Integers so:

val Y =fn : ((int→int) → (int→int)) → (int→int)

val F = fn : (int→int) →(int→int)

6-

Because of (2) and (+) the output is integer . so g(g) should be integer . thus g: int→int

Val f = fn : (int→int) →int

7-

It is totally similar to example in the book about reverse function.

We can infer the following type easily:

‘a list * ‘b → ‘b

However it is clear that it’s wrong since the output of function must be a list . this error occurs due to the fact that we have not got any “x” in the right side of second expression.

8-

It is totally similar to previous exercise. The output must be a value not a list . so it has to be ‘a . The error occurred since the wrong definition of REDUCE function . the correct version :

fun reduce(f, x::nil) = x | reduce(f, (x :: y)) = f(x, reduce(f, y));

9-

fun min (less , x :: nil) = x

| min (less , x :: xs) = if less(x,min(less,xs)) then x else min(less,xs)

Val min = fn : (‘a * ‘a → bool) * ‘a list → ‘a

The type was written based on function.

11-

a) fun atom (cons(a,b)) = nil
   | atom(symbol(“c”)) = symbol(“T”)
   | atom(Number(d)) = symbol(“T”)
   | atom(function(e)) = symbol(“T”)
b) fun isList(symbol(“a”) = nil
   | isList(Number(b)) = nil
   | isList(function(c)) = nil
   | isList(nil) = symbol(“T”)
   |isList(cons(e,f)) = isList(f)
c) fun car(nil) = nil
   | car(cons(a,b)) = a
d) fun lambda(x) = cons(x,symbol(“A”))