NAME _____

ML

Total 75 points

(2+3+4+4+5)

1. For each, write the type of a.

(a) fun a b:real = 5.0;

(b) fun a (b, c) = [b::c];

(c) fun a (b, c, d) = d=c(b);

(d) fun a b c = ("a"^b, c);

(e) fun a f L = remove_if f L; (* that *remove_if* you know *)

(3)

2. Convert the following expression to an equivalent expression without the *let* construct by replacing each *let* construct with function application form. Do **not** evaluate. Do **not** write any new definition.

```
let val x=1 and y=2 and z=3 in let val x=z+4 and y=x+5
in let val x=6 in (x,y,z) end end end
```

(3+3+3+4+4)

3. For each, write the value returned from sml.

(a)
```
- val x = 2;
- let val x = 3 and y = x+4 in let val x = x+5 and y = x+6
  in (x,y) end end;
```

(b)
```
- let val a = let val b = 2 in fn c => (b,c) end
  in let val b = 3 in a 4 end end;
```

(c)
```
- map (fn x => x::[x]) [[1,2,3], [4,5,6]];
```

(d)
```
- map (map (fn x => x::[x])) [[1,2,3], [4,5,6]];
```

(e)
```
- map (fn x => hd x) (map (fn x => x::[]) [[1,2,3], [4,5,6]]);
```

(2+3+5)

4.

(a) Considering an inductive definition of the set of powers of two, $\{ 2^n \mid n \in \mathbb{N} \}$, define datatype PowOfTwo to represent all the powers of two.

(b) Define function PowToInt to convert a power of two of type PowOfTwo to the corresponding integer.

(c) Define function which, for given two values of type PowOfTwo, returns their product of type PowOfTwo.

(5+7)

5.

(a) Define an **iterative** version of the following function $f : \mathbb{N} \rightarrow \mathbb{N}$.

$f(n) = n+1$ if $n<2$ and $f(n) = 3f(n-1) - f(n-2)$ otherwise

(b) Define a function that, given a list L, an object x, and a positive integer k, returns a copy of L with x inserted at the k-th position. e.g., $[a_1,a_2,a_3]$, x, and k=2 ==> $[a_1,x,a_2,a_3]$. If the length of L is less than k, insert at the end. Do **not** use *if-then-else*, *let*, *length*, or any auxiliary function.

(5+5+5)

6. The following must be **one-line definitions**. **No** *if-then-else* or *let*.

(a) Using both *map* and *reduce f L v*, define the function which computes the sum of all the integers in a given **list of lists of integers**.

(b) Using *reduce f L v* without *map*, define the function defined in (a).

(c) Using *reduce f L v*, define a function which, for x and a nonempty list of functions $[f_1, f_2, ..., f_k]$, returns $f_1(f_2(... (f_k(x)) ...))$, i.e., all the functions applied to x sequentially from the end.