p 13 q 10
GMB

1999

Introduction to Functional Programming

## Model Answer

1. Bookwork (slide 65).

```
fun foldl f (e,[]) = e
  | foldl f (e,x::xs) = foldl f (f(e,x),xs);
```

2. Bookwork (slide 65).

```
fun foldr f (e,[]) = e
  | foldr f (e,x::xs) = f(x,foldr f (e,xs));
```

3. (a) `fun append(xs,ys) = foldr (op::) (ys,xs);`
   (b) `fun length l = foldr (fn (x,c) => 1+c) (0,l);`
   (c) `fun map f l = foldr (fn (x,xs) => f(x)::xs) ([],l);`

4. Students were shown in the lectures that `foldl` and `foldr` can be visualised as

$$(\cdots((e \oplus x_1) \oplus x_2)\cdots)$$

$$(x_n \oplus \cdots (x_2 \oplus (x_1 \oplus e))\cdots)$$

respectively (this is the point of the hint). From this it is quite straightforward to see that if they are to be equal it is sufficient that the function $\oplus$ be associative and $e \oplus x_1 = x_1 \oplus e$. More formally, the theorem

`foldl f (e,xs)` $=$ `foldr f (e,xs)`

holds if `(f,e)` forms a *monoid* (students are only expected to give the two conditions), i.e.

   (a) `f` is associative, i.e. `f(f(a,b),c)=f(a,f(b,c))`.
   (b) `e` is a zero element, i.e. `f(a,e)=f(e,a)=a`.