Introduction to Functional Programming 2004
Paper 12 Question 10 (GMB)

(*a*) We are given the following code.

```
fun cat (b,f) nil     = b
  | cat (b,f) (x::xs) = f(x,(cat (b,f) xs));
```

The categorically-inclined reader will notice that `cat` defines a list catamorphism. The students have seen this function in the notes in a slightly different form, as the function `foldr`.

(*i*) `'a * ('b * 'a -> 'a) -> 'b list -> 'a`

(*ii*) `fun filter p = cat ([],(fn (x,xs) => if p(x) then x::xs else xs));`

(*iii*) `fun cmap f = cat ([],(fn (x,xs) => f(x)::xs));`

(*b*) We are given the following code.

```
fun ana (p,g) b = if p(b) then
                            []
                  else      let val (a,b1)=g(b)
                            in
                              a::(ana (p,g) b1)
                            end;
```

The well-read categorically-inclined reader will recognise that `ana` defines a list *anamorphism.*

(*i*) `('a -> bool) * ('a -> 'b * 'a) -> 'a -> 'b list`

(*ii*)

```
fun zip l = ana ((fn (a,b)         => (a=nil) orelse (b=nil)),
                 (fn (x::xs,y::ys) => ((x,y),(xs,ys)))
                ) l;
```

Note that one may have been tempted to define a value rather than a function, but this falls foul of the value polymorphism restriction of SML'97. No candidates would be penalised for this mistake.

(*iii*)

```
fun amap f = ana ((fn xs      => xs=nil),
                  (fn (x::xs) => (f(x),xs))
                 );
```