Exercise 1. Circle the free variables in the following code fragments.

```
local X in
    Y = Y + X
End

local F in
    X = {F {H 2} X*2}
End

fun {P X}
    if X =< 0 then X
    else {P (X-2)} end
end

local L in
    case L of
        nil then 0
        [] H | T then H
    end
end
```

Exercise 2. Consider a generic binary tree data structure of the following form:

```
<BTree A> ::=  nil  |  node(A, <BTree A>, <BTree A>)
```

Note that A denotes the generic type for each element of the tree. Using pattern-matching constructs and recursion, write Óz programs to perform the following whereby informal types have already been given.

i) A function that counts the number of elements in a given tree.
   // Count : <BTree A> Æ Int

ii) A function that returns a list of elements satisfying a given predicate.
    // FilterTree : {<BTree A>, (A Æ Bool)} Æ <List A>

iii) A function that partitions the elements of a tree into two lists based on a given predicate. Those elements satisfying the predicate are returned in the first list, and the rest are returned in the second list.
    // Partition : {<BTree A>, (A Æ Bool)} Æ <List A> # <List A>
**Question 3. Higher-Order Programs**

Consider the following higher-order functions:

```haskell
fun {FoldR F U L}
  case L of
    nil then U
    [] X|L2 then {F X {FoldR F U L2}}
  end
end

fun {Map F XS}
  case Xs of
    nil then nil
    [] X|Xr then {F X}|{Map F Xr}
  end
end
```

Predict the output (data structure being returned) for the following code fragments. If there is a program error, please describe it.

(i) `{Map (fun {$ X} X>3 end) [2 3 4 5] }`

(ii) `{Map (fun {$ X} X+3 end) [2 3 4 5] }`

(iii) `{FoldR (fun {$ X U} 1+U end) 0 [2 3 4 5] }`

(iv) `{FoldR (fun {$ X U} X end) 0 [2 3 4 5]}`

(v) `{FoldR (fun {$ X U} X end) 0 nil }`

(vi) `{FoldR (fun {$ X U} if X mod 2!=0 then X|U else U end end) nil [2 3 4 5]}

(vii) `{Map (fun {$ X} [X] end) [2 3 4 5] }`

(viii) `{Map (fun {$ X} 1.3 end) [2 3 4 5] }`

(ix) `{Map (fun {$ X} (fun {$ N} N+X end) end) [2 3 4 5] }`

(x) `{FoldR (fun {$ X U} U end) 0 [2 3 4 5]}