# 实验二

# 课堂练习(记录答案，需要检查)

**1. 分析以下函数或表达式的类型(先自己分析再程序验证)：**

```
fun all (your, base) =
        case your of
                0 => base
              | _ => "are belong to us" :: all(your - 1, base)

fun funny (f, []) = 0
| funny (f, x::xs) = f(x, funny(f, xs))

(fn x => (fn y => x)) "Hello, World!"
```

# 2. 参看ppt page 4-7的提示，用归纳法证明ins函数和isort函数的正确性

```
fun ins (x, [ ]) = [x]
   | ins (x, y::L) = case compare(x, y) of
                          GREATER => y::ins(x, L)
                        |     _   => x::y::L
```

对任一整数x和有序整数序列L，函数ins(x, L) 计算结果为x和L中所有元素构成的一个有序序列。

```
isort : int list -> int list

(* REQUIRES true                                  *)
(* ENSURES isort(L) = a sorted perm of L          *)

fun isort [ ] = [ ]
   |  isort (x::L) = ins (x, isort L)
```

对所有整数序列L，isort L计算得到L中所有元素的一个有序排列。

# insertion

ins : int * int list -> int list

::ins(x, L)

# proof outline

) = a sorted perm of x::A.

# insertion sort

isort : int list -> int list

ion of L.

# proof outline

# 3. 分析下面菲波拉契函数的时间复杂度 (O(n))

```
fun fib n = if n<=2 then 1 else fib(n-1) + fib(n-2);

fun fibber (0: int) : int * int = (1, 1)
  | fibber (n: int) : int * int =
      let val (x: int, y: int) = fibber (n-1)
      in (y, x + y)
      end
```

# 上机实验目标

- 掌握list结构的ML编程方法和程序性能分析方法
- 掌握基于树结构的ML编程方法和程序性能分析方法

# 内容：

1. 编写函数reverse和reverse’，要求：
   ① 函数类型均为：int list->int list，功能均为实现输出表参数的逆序输出；
   ② 函数reverse不能借助任何帮助函数；函数reverse’可以借助帮助函数，时间复杂度为$O$(n)。

2. 编写函数 interleave: int list * int list -> int list，该函数能实现两个int list数据的合并，且两个list中的元素在结果中交替出现，直至其中一个int list数据结束，而另一个int list数据中的剩余元素则直接附加至结果数据的尾部。如：

interleave([2],[4]) = [2,4]

interleave([2,3],[4,5]) = [2,4,3,5]

interleave([2,3],[4,5,6,7,8,9]) = [2,4,3,5,6,7,8,9]

interleave([2,3],[ ]) = [2,3]

3.编写函数listToTree: int list -> tree，将一个表转换成一棵平衡树。

提示：可调用split函数，split函数定义如下：

如果L非空，则存在L1, x, L2，满足：

split L = (L1, x, L2)　　且

L = L1 @ x :: L2　　且

length(L1)和length(L2)差值小于1。

4.编写函数revT: tree -> tree，对树进行反转，使trav(revT t) = reverse(trav t)。（trav为树的中序遍历函数）。假设输入参数为一棵平衡二叉树，验证程序的正确性，并分析该函数的执行性能（work和span）。

5. 编写函数binarySearch: tree * int -> bool。当输出参数1为有序树时，如果树中包含值为参数2的节点，则返回true；否则返回false。要求：程序中请使用函数Int.compare（系统提供），不要使用<, =, >。

```
datatype order = GREATER | EQUAL | LESS

case Int.compare(x1, x2) of
        GREATER => (* x1 > x2 *)
      | EQUAL => (* x1 = x2 *)
      | LESS => (* x1 < x2 *)
```