泛函程序设计原理

实验二

实验目的

- 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

实验内容：

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 *)
```