***2019***



**函数式编程 ·实验报告·**

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | ACM1701 |
| 学 号： | U201714780 |
| 姓 名： | 刘晨彦 |
| 完成日期： | 2020-04-07 |



目 录

[1 Lab 1 2](#_Toc37150363)

[1.1 代码实现 2](#_Toc37150364)

[1.2 代码测试 4](#_Toc37150365)

[2 Lab 2 6](#_Toc37150366)

[2.1 代码实现 6](#_Toc37150367)

[2.2 代码测试 9](#_Toc37150368)

[3 Lab 3 10](#_Toc37150369)

[3.1 代码实现 10](#_Toc37150370)

[3.2 代码测试 13](#_Toc37150371)

[4 实验心得 15](#_Toc37150372)

# Lab 1

## 代码实现

### 完成函数mult的编写，实现求解整数列表中所有整数的乘积

仿照sum函数，mult函数对传入参数进行匹配，若传入为空串，则返回1，若传入非空串，则将非空串首元素和剩余元素之积相乘。剩余元素之积通过mult函数递归实现。

具体实现代码如下图所示：

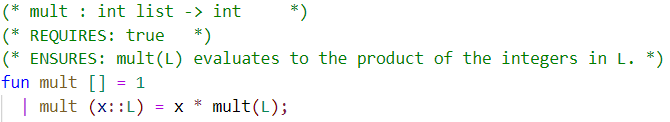


图 1.1 mult函数实现

### 完成如下函数Mult: int list list -> int的编写,该函数调用mult 实现int list list中所有整数乘积的求解

根据题目要求可知，传入Mult函数的参数是一个串的集合，需要对两种情况进行匹配：当传入为空串时，则返回1，若传入为非空串时，串首元素作为一个串，可以调用mult函数得到串元素的乘积，并乘以剩下串集中所有串元素的乘积，剩余串中所有串元素的乘积可以通过递归调用Mult函数得到。

具体实现代码如下图所示：

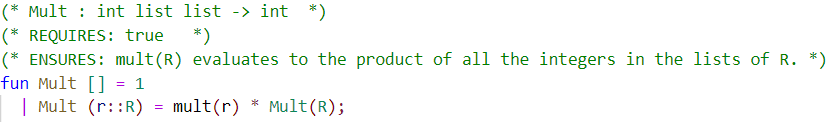


图 1.2 Mult函数实现

### 函数mult’定义如下，试补充其函数说明，指出该函数的功能。利用mult’定义函数Mult’，使对任意整数列表的列表R和整数a，该函数用于计算a与列表R中所有整数的乘积

已知mult’的定义如下所示：

fun mult’ ([ ], a) = a

| mult’ (x :: L, a) = mult’ (L, x \* a);

根据函数可知，mult’将串L的每个元素都和a相乘，当L为空串时得到a。故函数的功能是得到串L内每个元素和a的积。补充函数说明如下图所示：

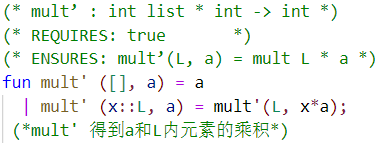


图 1.3 mult’的函数说明和功能

要实现Mult’，则可以利用Mult函数来计算。实现如下图所示：

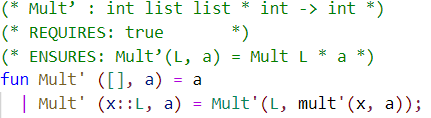


图 1.4 Mult’函数的实现

### 编写递归函数square实现整数平方的计算，要求：程序中可调用函数double，但不能使用整数乘法（\*）运算

根据题目可知，不能使用乘法运算，因此需要利用double来实现。根据等式，即可利用递归实现。代码实现如下图所示：

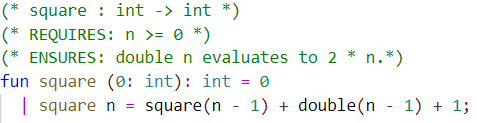


图 1.5 square函数的实现

### 定义函数divisibleByThree: int -> bool，以使当n为3的倍数时，divisibleByThree n为true，否则为false。注意：程序中不能使用取余函数’mod’

根据要求，可以利用递归，每次对传入的n减去3，直到，仅当n=0时得到true，其余情况得到false。函数实现如下图所示：

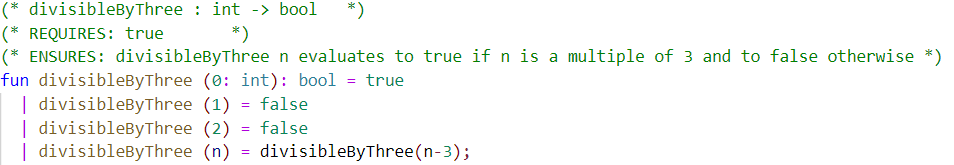


图 1.6 divisibleByThree函数的实现

### 编写奇数判断函数oddP: int -> bool，当且仅当该数为奇数时返回true

函数实现方法和divisibleByThree相似，通过对大于等于2的n减去2并递归，直到n小于2，并判断是否为0，为0则为偶数。具体实现如下图所示：

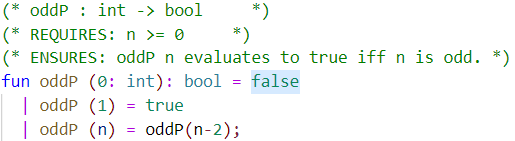


图 1.7 oddP函数的实现

## 代码测试

编写测试代码如下图所示：

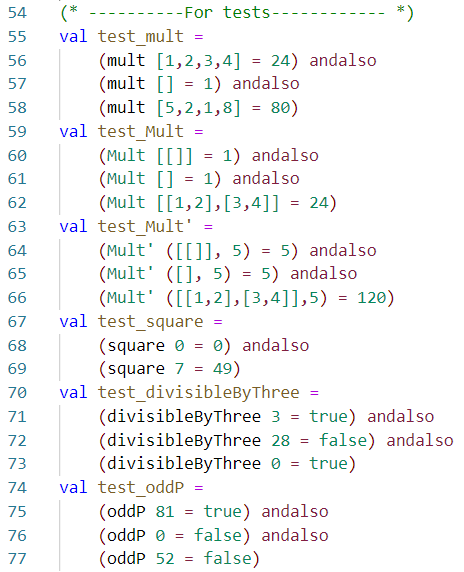


图 1.8 Lab1测试代码

测试结果如下图所示：

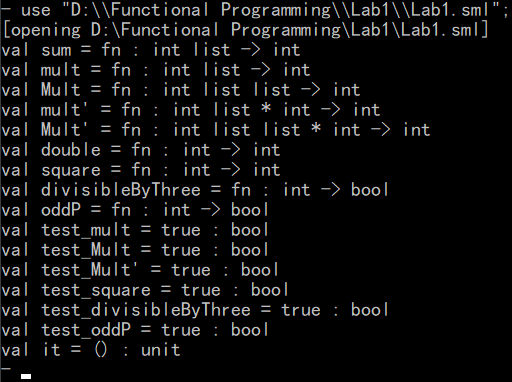


图 1.9 Lab1 测试情况

根据测试结果可知，代码功能正确。

# Lab 2

## 代码实现

### 编写函数reverse和reverse’，要求：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；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)

对reverse函数，若传入为空串，则得到空串，若传入非空串，则将首元素接在剩余串逆序后的尾部即可，代码实现如下图：

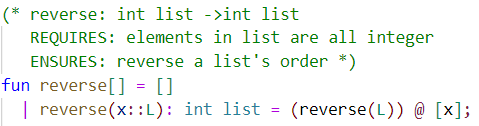


图 2.1 reverse函数实现

对reverse’函数，则可以通过辅助函数，每次将输入串L中的首元素压入新串R，最后得到R。具体实现如下图所示：

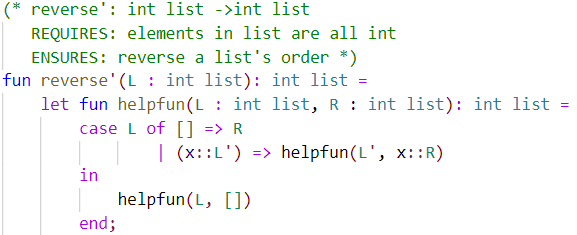


图 2.2 reverse’函数实现

### 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

根据题目要求，匹配有四种情况：两串均为空串，则直接返回组装好的新串；若其中一个串为空串，一个为非空串，则将非空串接在新串后面。若两串均为非空串，则将两非空串的首元素取出并接在新串尾部。新串初始化为空串。具体实现如下图所示：

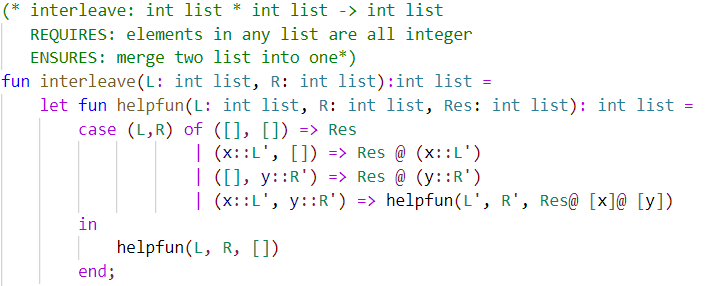


图 2.3 interleave 函数的实现

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

函数的实现方式是，将串根据长度一分为二，将中间的元素作为当前树根，并对左右两串递归操作，得到左右子树。具体实现如下图所示：

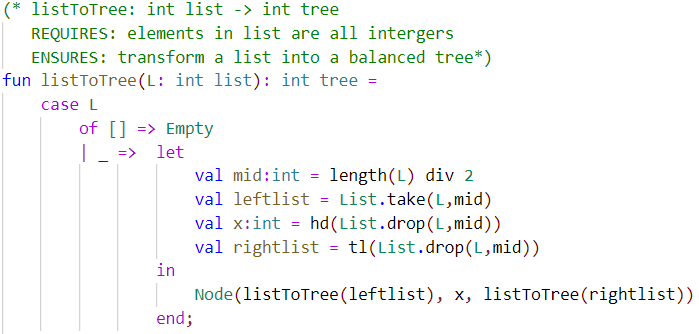


图 2.4 listToTree函数的实现

### 编写函数revT: tree -> tree，对树进行反转，使trav(revT t) = reverse(trav t)。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）

对于输入的平衡二叉树，对每个Node，交换左右子树位置，并对左右子树重复该操作。代码具体实现如下图：

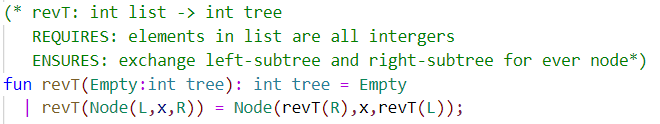


图 2.5 revT 函数的实现

假设树的节点数（各子树根节点与叶子节点之和）为n，则平衡树树高为，认为在每一层的交换位置都是并行的，故span为：。每一层i中，有work：，故总的work为：。

### 编写函数binarySearch: tree \* int -> bool。当输入参数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<, =, >

对输入的树，当为空节点时，得到false，当不为空时，比较当前根节点和参数2的大小，若根节点较小，则在右子树中递归，若相同，则返回true，若更大，则在左子树中递归寻找。代码实现如下：

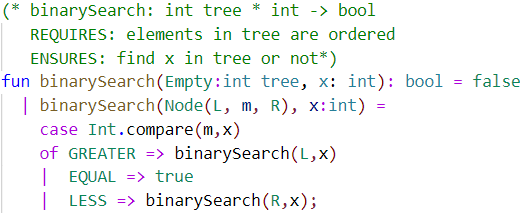


图 2.6 binarySearch 函数的实现

## 代码测试

编写测试程序如下图所示：

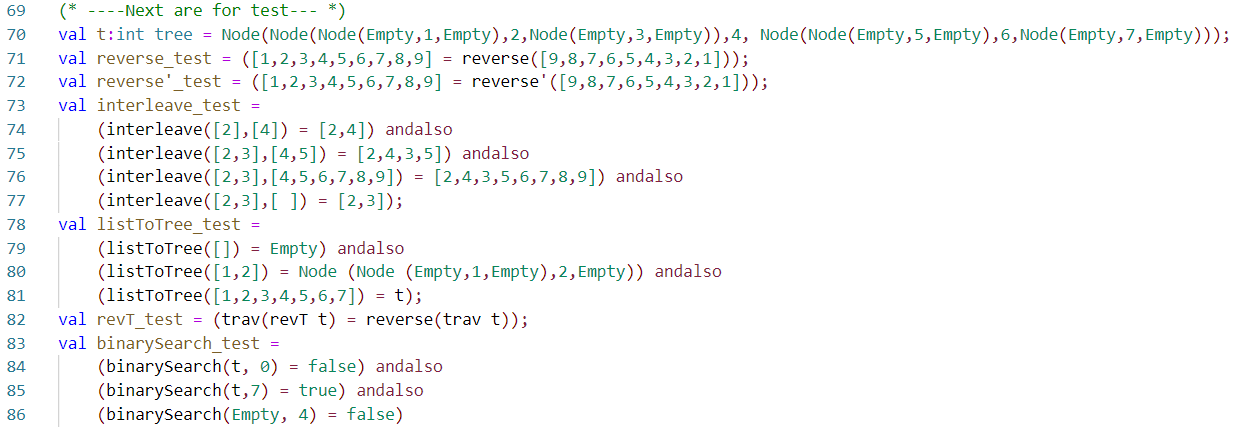


图 2.7 Lab2 测试代码

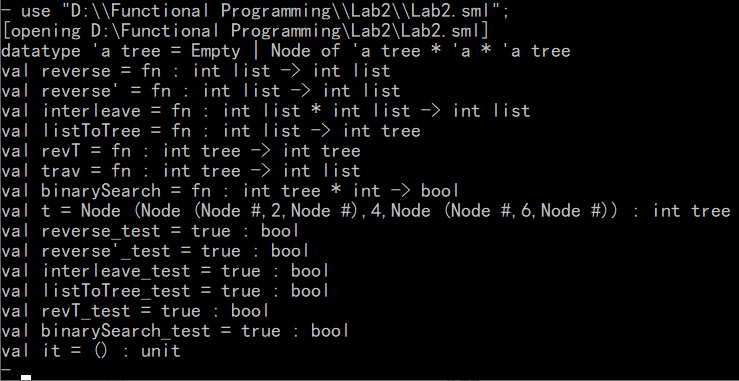


图 2.8 Lab2 测试结果

根据测试情况可知，Lab2代码功能正确。

# Lab 3

## 代码实现

### 编写函数thenAddOne，类型为: ((int ->int) \* int) -> int；功能为将一个整数通过函数变换(如翻倍、求平方或求阶乘)后再加1

根据函数类型可知，thenAddOne有两个参数：一个函数和一个整数。根据函数功能可知，首先对整数进行函数计算，然后计算结果加一。具体实现如下图：

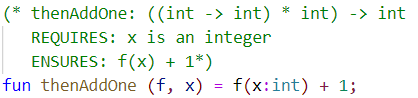


图 3.1 thenAddOne 函数的实现

### 编写函数mapList，函数类型为: ((‘a -> ‘b) \* ‘a list) -> ‘b list；功能为实现整数集的数学变换(如翻倍、求平方或求阶乘)

根据函数类型可知mapList有两个参数：一个函数和一个串。根据函数功能可知，需要依次从串中取出元素做函数操作，并组成新串。函数实现如下图所示：

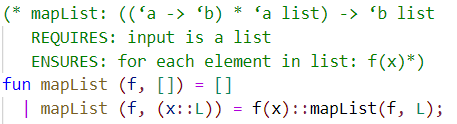


图 3.2 mapList函数的实现

### 编写函数mapList’，函数类型为: (‘a -> ‘b) -> (‘a list -> ‘b list)，功能为实现整数集的数学变换(如翻倍、求平方或求阶乘)；比较函数mapList’和mapList，分析、体会它们有什么不同

根据题目可知，mapList’是一个多态函数，其本身计算的结果就是一个函数。对于mapList’，同样需要依次从串中取出元素做函数操作，并组成新串。函数实现如下图所示：

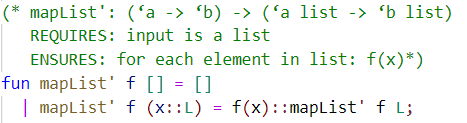


图 3.3 mapList’函数的实现

### 编写函数findOdd，要求：函数类型为: int list -> int option；功能为：如果x为L中的第一个奇数，则返回SOME x；否则返回NONE

根据题目要求，依次取出串中首元素进行判断是否为基数，若是返回SOME x，否则递归检查下一个元素，直至串被取空，返回NONE。具体实现如下图所示。

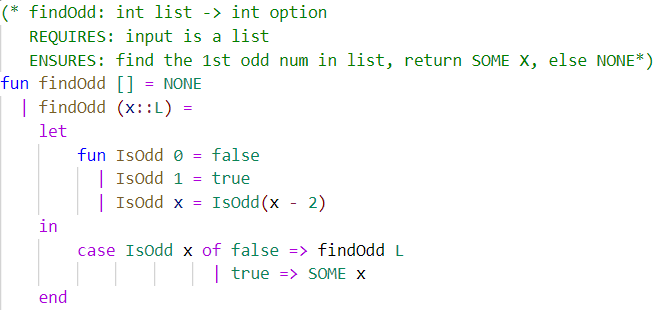


图 3.4 findOdd函数的实现

### 编写函数subsetSumOption: int list \* int -> int list option，要求：对函数subsetSumOption(L, s)：如果L中存在子集L’，满足其中所有元素之和为s，则结果为SOME L’；否则结果为NONE

对于传入的串L，首先使用subset函数，得到类型为int list list的子串集合。对该集合，每次取出一个子集L’，计算子集元素之和，与参数2进行大小比较，若相同则返回SOME L’，若不同则递归比较下一个子集，直至没有子集剩余，返回NONE。具体实现如下图：

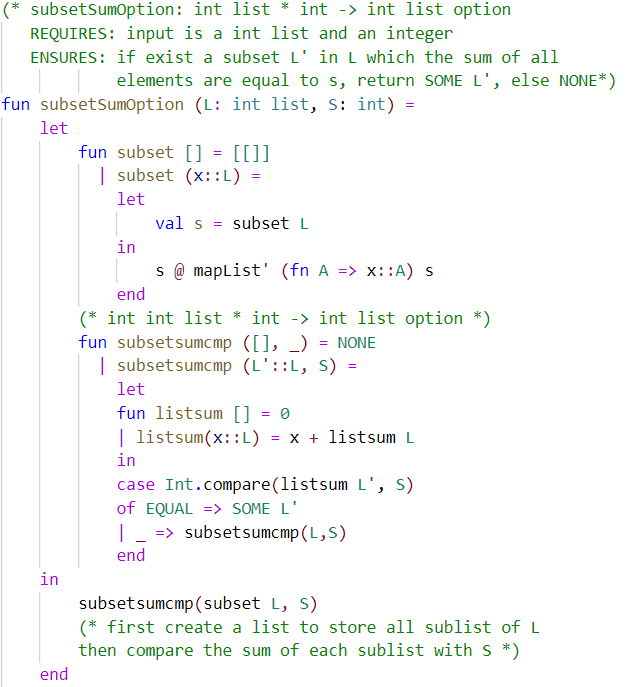


图 3.5 subsetSumOption函数的实现

### 编写函数：exists: (‘a -> bool) -> ‘a list -> bool；forall: (‘a -> bool) -> ‘a list -> bool，对函数p: t -> bool, 整数集L: t list,有：exist p L =>\* true if there is an x in L such that p x=true；exits p L =>\* false otherwise； forall p L =>\* true if p x = true for every item x in L；forall p L =>\* false otherwise

对于exists函数，每次从list中取出首元素比较是否满足要求，若满足则返回true，否则递归检查下一个首元素，直至list为空，返回false。

对于forall函数，每次从list中取出首元素比较是否满足要求，若不满足则直接返回false，否则递归检查下一个首元素，直到list为空，返回true。两个函数的具体实现如下图：

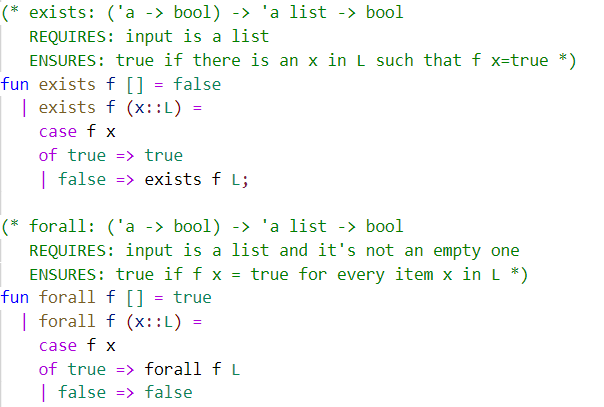


图 3.6 exists函数和forall函数的实现

### 编写函数：treeFilter: (‘a -> bool) -> ‘a tree -> ‘a option tree，将树中满足条件P（ ‘a -> bool ）的节点封装成option类型保留，否则替换成NONE

对传入的节点，若为空，则仍然得到空，若非空，则比较根节点x是否满足函数f，若满足则将该节点的根节点修改为SOME x，否则修改为NONE。对左右子树重复该操作。具体代码实现如下图：

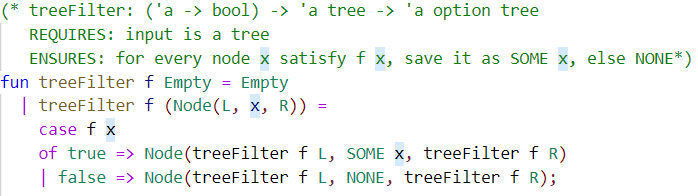


图 3.7 treeFilter函数的实现

## 代码测试

编写测试代码如下图所示：

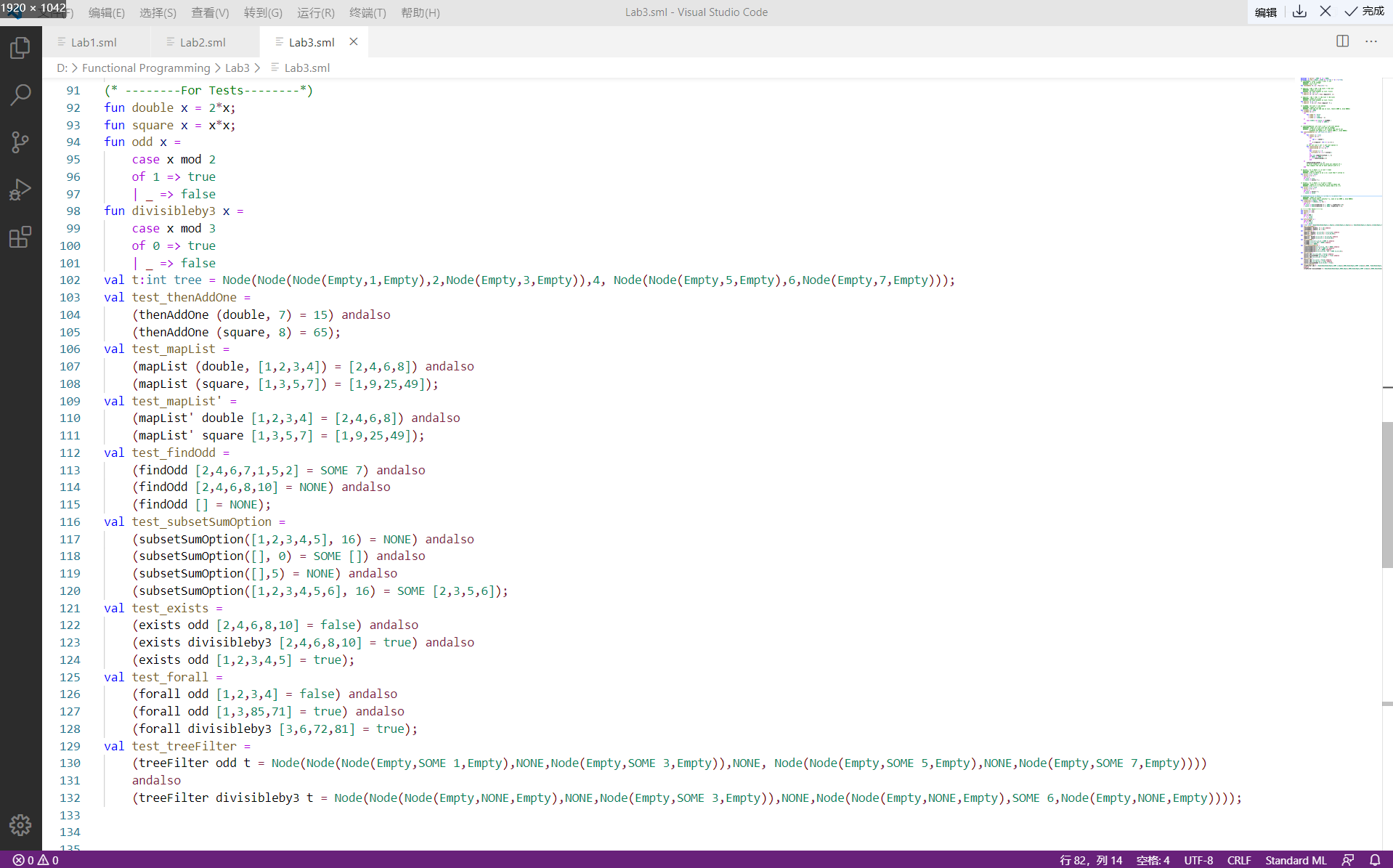


图 3.8 Lab3 测试代码

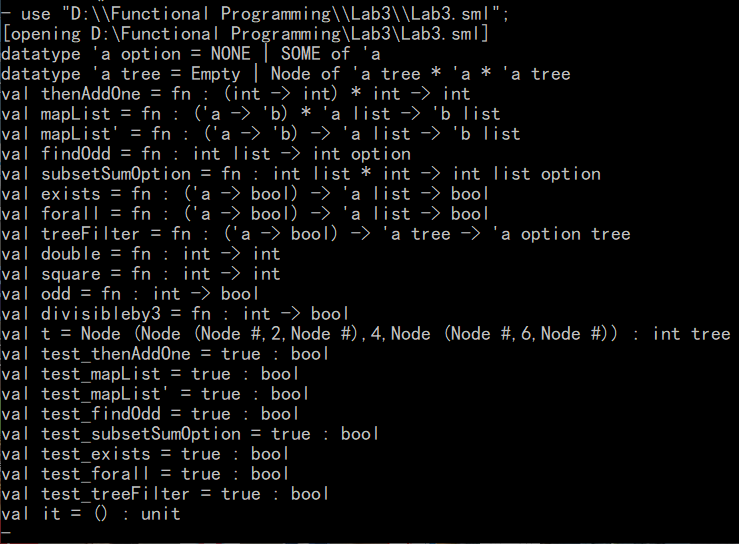


图 3.9 Lab3 测试结果

根据测试情况可知，代码功能正确。

# 实验心得

由于ACM班课程的原因，在大二上学期已经使用过sml语言，但是由于当时时间比较紧迫，对于sml语言的掌握只能说是一知半解，只求实现，不求得心应手。甚至由于当时的实验难度，一度对函数式编程感到畏惧。

但是在本学期选修函数式编程之后，终于有机会重新了解这门语言。通过回顾之前使用sml语言的经历，结合课上教授和课后实验的锻炼，终于克服了对于函数式编程的畏惧，并且能够熟练掌握函数式编程的技巧和思维方式，收获匪浅。

三次实验侧重各有不同，第一次实验主要侧重于熟练函数式编程的递归精神，第二次实验侧重于树结构，第三次实验侧重于多态函数和高阶函数，三次实验虽然难度不高，仅有几题稍要思考，但对提高学生面对陌生编程方式的信心大有裨益，同时也锻炼了学生函数式编程的能力。