第二次作业 返回

姓名：熊逸钦 班级：计算机1701-4 成绩： 96分

一.简答题 （共6题,100.0分）

1

下列模式能否与类型为int list的L匹配成功？如果匹配不成功，指出该模式的类型？（假设x为int类型）

x::L 非空list

_::_ 非空list

x::(y::L)

(x::y)::L

[x, y]

正确答案：

我的答案：
1. 成功
2. 成功
3. 当list成员大于等于2时，成功
4. 错误，匹配的是int list list
5. 当list成员等于2时，成功

2 试写出与下列表述相对应的模式。如果没有模式与其对应，试说明原因。

list of length 3

lists of length 2 or 3

Non-empty lists of pairs

Pairs with both components being non-empty lists

正确答案：

我的答案：
[x,y,z]
没有，可以拆分成[x,y]和[x,y,z]两个模式分别进行描述
(x,y)::L
(x::L1, y::L2)

3 分析下述程序段（左边括号内为标注的行号）：

```
(1)     val x : int = 3
(2)     val temp : int = x + 1
(3)     fun assemble (x : int, y : real) : int =
(4)         let val g : real = let val x : int = 2
(5)                            val m : real = 6.2 * (real x)
```

```
(6)                                 val x : int = 9001
(7)                                 val y : real = m * y
(8)                             in y – m
(9)                             end
(10)          in
(11)            x + (trunc g)
(12)          end
(13)□(14)     val z = assemble (x, 3.0)
```

试问：第4行中的x、第5行中的m和第6行中的x的声明绑定的类型和值分别为什么？第14行表达式assemble(x, 3.0)计算的结果是什么?

正确答案：

我的答案：
第4行的x：int，2
第5行的m：real，12.4
第6行的x：int，9001
第14行的计算结果：27

4 编写函数实现下列功能：

（1）zip: string list * int list -> (string * int) list

其功能是提取第一个string list中的第i个元素和第二个int list中的第i个元素组成结果list中的第i个二元组。如果两个list的长度不同，则结果的长度为两个参数list长度的最小值。

（2）unzip: (string * int) list -> string list * int list

其功能是执行zip函数的反向操作，将二元组list中的元素分解成两个list，第一个list中的元素为参数中二元组的第一个元素的list，第二个list中的元素为参数中二元组的第二个元素的list。

对所有元素L1: string list和L2: int list，unzip(zip (L1, L2)) = (L1, L2)是否成立？如果成立，试证明之；否则说明原因。

正确答案：

我的答案：

（1）

```
fun zip([],_) = []
  | zip(_,[]) = []
  | zip(s::SL,x::IL) = (s,x)::zip(SL,IL);
```

（2）

```
fun unzip([]) = ([],[])
  | unzip((s,x)::L) =
  let   val (SL,IL) = unzip(L)
  in    (s::SL,x::IL)
  end;
```

（3）不成立，因为L1的长度和L2可能不同。若L1=["a","b","c","d","e"]，L2=[1,2,3,4]，则unzip(zip (L1, L2))的结果为(["a","b","c","d"], [1,2,3,4])，可以看到["a","b","c","d"]和L1不等。

5

指出下列代码的错误：

```
(* pi: real *)
val pi : real = 3.14159;

(* fact: int -> int *)
fun fact (0 : int) : int = 1
  | fact n = n * (fact (n - 1));

(* f : int -> int *)
fun f (3 : int) : int = 9
    f _ = 4;
(* circ : real -> real *)
fun circ (r : real) : real = 2 * pi * r

(* semicirc : real -> real *)
fun semicirc : real = pie * r

(* area : real -> real *)
fun area (r : int) : real = pi * r * r
```

正确答案:

我的答案:
函数f的第二行开头漏了一个或符号 '|',
函数circ、semicirc和area末尾都漏了一个分号 ';',
函数circ中2是int类型，应该写成2.0的real类型，
函数semicirc没有匹配参数r，而且把pi写成了pie，应该写成"fun semicirc(r : real) : real = pi * r;",
函数area的参数r类型错误，应该写为real。

6 分析下面菲波拉契函数的执行性能

```
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
```

借助：对所有非负整数k，

fib(2k) = fib(k)(2fib(k + 1) − fib(k))

fib(2k + 1) = fib(k + 1)2 + fib(k)2

正确答案:

我的答案:
在函数fib中，每次递归调用需要展开到两个函数，时间复杂度为O(2^n)，
在函数fibber中，若输入值为n，则函数迭代n次即可得出结果，时间复杂度为O(n)。