作业一

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）

模式	结果	模式类型
x::L	成功	
_::_	成功	
x::(y::L)	成功	
(x::y)::L	不成功	int list list
[x,y]	不成功	[int * int]

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。

表述	模式
list of length 3	x::y::z::[]
lists of length 2 or 3	x::y::z::[] | x::y::[]
Non-empty lists of pairs	(x::L, _) | (_, y::R)
Pairs with both components being non-empty lists	(x::L, y::R)

3.分析下述程序段（左边括号内为标注的行号）,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？

```
(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)
```

变量位置	类型	值
第4行中的x	int	2
第5行中的m	real	12.4
第6行中的x	int	9001
第14行表达式assemble(x, 3.0)	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。

代码实现见下图，代码文件另附。

```
1   (* zip: string list * int list -> (string * int) list*)
2   fun zip ([], _): (string * int) list = []
3     | zip (_, []) = []
4     | zip (s::S, x::L) = (s,x)::zip(S,L);
5
6   val result = zip (["hello","world","cute"],[1,2,3,4]);
7
8   (* unzip: (string * int) list -> string list * int list *)
9   fun unzip (L: (string * int) list): string list * int list =
10      let
11          fun helpfun ([]:(string * int) list, slist:string list, intlist:int list) = (slist, intlist)
12            | helpfun (x::L, slist, intlist) = helpfun(L,slist@[#1 x], intlist@[#2 x])
13      in
14          helpfun(L,[],[])
15      end;
```

对于所有元素L1: string list和L2: int list，unzip(zip (L1, L2)) = (L1, L2)不一定成立，因为zip函数会省略较长串中超过较短串的部分。仅当两串长度相同时成立。

5.指出下列代码错误

```
(* 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
```

- circ函数：运算符*两侧的类型需要相同，而数字2的类型为整数型，应该改为2.0；
- semicirc函数：运算参数pie和r均未传入。根据函数功能，需要将r传入，将pie改为pi
- area函数：运算符*两侧类型需要相同，故将x转换为real类型
- f函数：类型匹配需要加 |

6.分析下面斐波那契函数的执行性能

```
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
```

fib函数work：$\because T(n) = T(n-1) + T(n-2) + O(1),\ \therefore T(n) = O(2^n)$，fib函数$\text{span} = O(n)$

fibber函数work：$\because T(n) = T(n-1) + O(1),\ \therefore T(n) = O(n), \text{span} = O(n)$